DASTE: Decentralized Ad-Hoc Access-Structured Threshold Encryption with Dynamic Policy Evolution

Anil Kumar Pradhan[1] and Abhraneel Dutta[2]

[1] Vaultree Ltd, Ireland
anil@vaultree.com
[2] Florida International University, USA
abdutta@fiu.edu

Abstract. We introduce *DASTE*, a decentralized encryption primitive for auditable access control in settings where users independently generate public keys, register them on an immutable ledger, and decrypt only through *collaboration*. In DASTE, a sender encrypts under an access structure (e.g., an access tree / LSSS) whose leaves are *concrete registered public keys selected at encryption time*. A ciphertext can therefore be opened only by a qualifying coalition of registered key holders that jointly reconstructs the masking secret.
DASTE is designed for dynamic policy-governed environments in which access conditions may need to change after encryption. To support this, we provide *ciphertext-only policy evolution* operations, including semantically neutral insertion, threshold escalation, subtree revocation, and ciphertext rerandomization, that update ciphertexts without reissuing user secret keys and without requiring plaintext access. We give two instantiations: a classical discrete-log-based construction, included as a conceptual baseline, and a post-quantum construction based on decisional Ring-LWE. For the RLWE construction, we prove coalition-bounded IND-CPA security via a standard hybrid argument. Together, these results yield a ledger-anchored, access-structured, post-quantum threshold encryption framework suitable for decentralized key management and governance-oriented decryption workflows.

Keywords: Ad-hoc threshold encryption, Attribute-based Encryption, Fully Homomorphic Encryption (FHE), Post-Quantum Cryptography

1 Introduction

As decentralized infrastructures (consortium ledgers, cross-organization clouds, and self-sovereign identity ecosystems) become the default substrate for inter-organizational collaboration, access control to encrypted data is increasingly forced to satisfy two competing demands: *(i)* it must be *auditable and governance-friendly* (who was eligible to decrypt, and under what policy?), and *(ii)* it must

be *cryptographically future-proof* in the face of the ongoing post-quantum migration toward standardized lattice-based primitives [15, 16].

A natural abstraction for this setting is *ad-hoc threshold public-key encryption* (TPKE), where a sender selects recipients and an access condition at encryption time, and decryption requires a qualifying coalition. Dynamic TPKE and related ad-hoc threshold/broadcast encryption schemes support such sender-chosen groups [4, 5], but typically rely on trusted setup, master secret state, or enrollment procedures. Silent-setup TPKE removes the trusted dealer by allowing users to generate keys locally [9], and recent work extends this direction toward more expressive distributed policy encryption under pairing-based assumptions [6, 18]. In parallel, registration- and directory-based cryptography replaces centralized key issuance with public registries [7, 8], while transparency systems such as Certificate Transparency and CONIKS provide auditable key-registration mechanisms [12, 14]. More recent lattice-based constructions begin to address post-quantum security for trustless-directory encryption [3, 19].

However, decentralized, ad-hoc, coalition-based access control still lacks three features in combination. First, existing ad-hoc and silent-setup TPKE schemes largely treat policies as ciphertext-rigid: strengthening a threshold or pruning members generally requires re-encryption rather than ciphertext-only updates [5,9]. Second, efficient dealerless constructions often rely on pairings, which are unsuitable for post-quantum deployment [9,18]. Third, ad-hoc TPKE mainly supports flat thresholds, while expressive policy encryption remains separate from decentralized key generation and ciphertext-side policy evolution [6, 18]. Although ABE and registered ABE provide expressive access structures and reduced authority dependence [1,10,11,13], they target single-user decryption and anti-collusion, whereas our setting requires coalition decryption over concrete registered keys selected at encryption time.

Contributions. We introduce *DASTE: Decentralized Ad-Hoc Access-Structured Threshold Encryption with Dynamic Policy Evolution*. DASTE is designed for decentralized ledger-based key management in which users independently generate public keys, authenticate them for registration (e.g., via standardized post-quantum signatures), and ciphertexts are bound to an access policy defined at encryption time that contains a concrete set of registered public keys and an access structure. Our main contributions are:

- **Access-structured ad-hoc threshold encryption over registered keys.** We formalize and construct a coalition-decryption encryption scheme where the sender selects a set of registered public keys and an access tree/LSSS policy at encryption time; a qualifying coalition reconstructs the masking secret to decrypt.
- **Ciphertext-only dynamic policy evolution.** We provide a suite of public update operations that enable post-deployment access-policy up-gradation without reissuing user keys or accessing plaintext: trivial gate insertion, increasing threshold, revoking access, and rerandomization for unlinkability.
- **Post-quantum instantiation from RLWE with IND-CPA security.** We instantiate DASTE from the decisional Ring-LWE assumption and prove

IND-CPA security, thereby achieving a dealerless, ad-hoc, access-structured threshold encryption scheme aligned with post-quantum migration requirements.

Compared with the shorter version, this extended version also includes the classical discrete-log instantiation, its correctness argument, and additional implementation level discussion of noise and gadget decomposition.

Our framework removes trusted authorities, PKGs, key curators, and master secret keys by letting users independently generate their secret keys and publish only signed public data to an immutable ledger, unlike CP-ABE and RABE/MA-RABE models that rely on central or partially trusted authority components [1, 10, 11, 13]. This eliminates key escrow and conventional key-distribution channels while preserving transparent verification, auditability, and tamper-evident revocation. DASTE further supports ciphertext-only policy evolution without key reissue or helper mechanisms, improving flexibility over prior access-control approaches [10,11]. Finally, our post-quantum instantiation relies on decisional Ring-LWE to provide IND-CPA security against quantum adversaries while retaining fine-grained, adaptive access control [17].

1.1 Roadmap

We begin with notation and preliminaries in Section 2. Section 3.1 formalises the system and threat model. Sections 3.2 and 3.3 detail the classical and post-quantum instantiations, respectively. Section 3.5 presents ciphertext-side policy-update algorithms, Section 4 describes detailed security analysis, and Section 5 reports a comparison.

2 Preliminaries

2.1 Notation

Let λ represent the security parameter, and let p and q be distinct large primes defining the finite fields $\mathbb{Z}_p$ and $\mathbb{Z}_q$, respectively. The classical scheme operates over a cyclic group $\mathbb{G}$ of prime order p with a generator g, and a bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is employed in the ABE construction, where $\mathbb{G}_T$ is the target group. In the post-quantum instantiation, computations are performed over the cyclotomic ring $R_q = \mathbb{Z}_q[x]/(x^N + 1)$, with N being a power of two and χ denoting a small, centered discrete Gaussian error distribution. The plaintext space is defined by the ring $R_p = \mathbb{Z}_p[x]/(x^N + 1)$.

2.2 KP-ABE, LSSS, and Extended Shamir Sharing

KP-ABE of Goyal et al. *[10].* Goyal *et al.* introduced a key-policy ABE scheme over bilinear groups. Let $\mathbb{G}_1$ be a cyclic group of prime order p with generator g, and let

$$e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$$

be a bilinear map. Setup samples $t_1, \ldots, t_n, y \xleftarrow{\$} \mathbb{Z}_p$, publishes

$$\mathrm{PK} = \big(\{T_i = g^{t_i}\}_{i=1}^{n},\ Y = e(g,g)^{y}\big),$$

and keeps $\mathrm{MK} = (t_1, \ldots, t_n, y)$. To encrypt $M \in \mathbb{G}_2$ under an attribute set $\gamma \subseteq \{1, \ldots, n\}$, the encryptor samples $s \xleftarrow{\$} \mathbb{Z}_p$ and outputs

$$E = \big(\gamma,\ E' = M \cdot Y^{s},\ \{E_i = T_i^{s}\}_{i\in\gamma}\big).$$

A secret key for an access tree $\mathcal{T}$ is generated by assigning to each node x a polynomial q_x of degree $k_x - 1$, with root value $q_r(0) = y$ and child consistency $q_x(0) = q_z(\mathsf{index}(x))$. For a leaf x labeled by attribute i, the key component is

$$D_x = g^{q_x(0)/t_i}.$$

Decryption is recursive: a satisfied leaf computes

$$e(D_x, E_i) = e(g,g)^{sq_x(0)},$$

and internal nodes combine child values via Lagrange interpolation. At the root, one obtains $Y^s = e(g,g)^{ys}$, which cancels the mask in E' and recovers M. The scheme is selectively secure under standard bilinear assumptions [10].

LSSS access trees. An LSSS access structure represents a policy as a rooted tree whose internal nodes are threshold gates and whose leaves are labeled by attributes. A (k, n)-gate is satisfied when at least k of its n children are satisfied. During encryption, a secret is shared recursively from the root to the leaves using node polynomials; decryption reconstructs the secret from any satisfying set of leaves using Lagrange interpolation. This gives a compact way to express monotone, fine-grained access policies [10].

Extended Shamir secret sharing over R_q. Let

$$R_q = \mathbb{Z}_q[x]/(x^N + 1),$$

let $P = \{P_1, \ldots, P_n\}$ be the participants, let $t \le n$ be the threshold, and let $s \in R_q$ be the secret. To split s, sample a random degree-$(t-1)$ polynomial

$$d(y) = s + \sum_{j=1}^{t-1} d_j y^j, \qquad d_j \in R_q,$$

choose distinct nonzero points $y_1, \ldots, y_n \in \mathbb{Z}_q^*$, and define shares

$$s_i = d(y_i) \in R_q.$$

Any qualified set $S \subseteq \{1, \ldots, n\}$ of size t reconstructs the secret using Lagrange coefficients

$$L_{i,S}(0) = \prod_{\substack{j\in S \\ j\neq i}} \frac{0 - y_j}{y_i - y_j}, \qquad s = \sum_{i\in S} s_i\, L_{i,S}(0).$$

We denote these procedures by $\textbf{ESSS.Split}(R_q, n, t, s)$ and $\textbf{ESSS.Reconstruct}(R_q, \{(y_i, s_i)\}_{i\in S})$. This is the ring-based threshold-sharing primitive used in the post-quantum construction.

2.3 The Decisional Ring-LWE Problem

Let N be a power of two, q a prime with $q \equiv 1 \pmod{2N}$, and $R_q = \mathbb{Z}_q[x]/(x^N + 1)$ the degree-N cyclotomic ring. Fix a noise distribution χ over R_q (e.g. a centred binomial or discrete Gaussian) with small standard deviation. For a secret $s \xleftarrow{\$} \chi$ and independently sampled $a \xleftarrow{\$} R_q$, $e \xleftarrow{\$} \chi$, an *RLWE sample* is the pair

$$(a,\ b = a \cdot s + e) \ \in \ R_q \times R_q. \tag{1}$$

The *decisional RLWE* problem asks to distinguish polynomially many independent RLWE samples from uniformly random pairs in $R_q \times R_q$. Formally, the advantage of a probabilistic polynomial-time distinguisher $\mathcal{D}$ is

$$\mathrm{Adv}_{\mathcal{D}}^{\mathrm{RLWE}} \ = \ \Big|\Pr\big[\mathcal{D}(a,b) = 1 \mid b = a \cdot s + e\big] \ - \ \Pr\big[\mathcal{D}(a,u) = 1 \mid u \xleftarrow{\$} R_q\big]\Big|. \tag{2}$$

The *Decisional RLWE Assumption* states that for any PPT $\mathcal{D}$ this advantage is negligible in the security parameter [17]. This assumption underlies the semantic security of our post-quantum DASTE instantiation.

Definition 1 (Discrete Logarithm). *Given a pair $(g,\ g^x) \in \mathbb{G}^2$ sampled with $x \xleftarrow{\$} \mathbb{Z}_p$, no probabilistic polynomial–time (PPT) adversary can output x with non–negligible probability in λ.*

2.4 Gadget Decomposition in Fully Homomorphic Encryption

Gadget decomposition is a technique used in lattice-based Fully Homomorphic Encryption (FHE) to manage noise growth during homomorphic operations. Given a polynomial $z \in R_q = \mathbb{Z}_q[x]/(x^N + 1)$ with q a modulus and N a power of two, we decompose z into a base-w representation, where w is a gadget base and $\ell = \lceil \log_w q \rceil$ is the number of levels. This yields $z = \sum_{l=0}^{\ell-1} z_l w^l$, with coefficients $z_l \in \mathbb{Z}$ satisfying $|z_l| < w$. Direct multiplication of z with a ciphertext component $T_a = \alpha t_a + p e_a$ will increase the noise from e_a to $z.e_a$ which is equivalent to $\mathcal{O}(N.q.e_a)$ and will corrupt the underlying plaintext making it impossible to decrypt. To keep the accumulated error within decryptable bounds, we employ a gadget-style decomposition of each attribute public key, inspired by techniques introduced in lattice-based FHE schemes [2].

3 Construction

This section presents the construction of *DASTE*, our decentralized, ledger-anchored *ad-hoc access-structured threshold encryption* scheme. First we describe the system model and then the actual construction.

At a high level, a sender samples a masking secret, shares it over an access tree, and binds each leaf share to a *concrete public key registration* selected from the ledger. Decryption is intentionally collaborative: each participating user locally opens its own leaf share, and a qualifying coalition reconstructs the masking secret via LSSS interpolation.

3.1 System Model

DASTE involves three primary entities:

- **Users** independently generate and register public keys, and may later contribute local decryption shares;
- **Data owners** encrypt messages under explicit access structures over registered keys;
- An **immutable public ledger** (e.g. a blockchain) stores global parameters, signed key registrations, and optionally signed policy commitments.

A decryption request may additionally involve a *combiner* (e.g. one of the users or the requesting application) that aggregates valid local shares and performs the final reconstruction; this is a convenience role only and is *not* trusted with any master secret.

The workflow proceeds in five phases.

1. *System initialisation.* An initial distributed setup protocol, publishes the public parameters PP defining the algebraic setting for both the classical (DLP/EC-DLP) and post-quantum (RLWE) instantiations. No master secret is generated, and no party obtains escrow capability.
2. *User key generation and registration.* Each user independently samples its own secret key and derives one or more *public keys* associated with access labels and *public identifiers*. Each public key registration, together with its associated label and signature, is written to the distributed ledger.
3. *Access-structure/Policy definition.* A data owner defines an access structure $\mathcal{P}$, represented as a monotone Boolean formula or equivalently as an LSSS access tree (T, τ). Crucially, the leaves of T refer to *published public identifiers* corresponding to users selected at encryption time. A policy identifier or digest may also be committed to the ledger.
4. *Data encryption.* To encrypt a message M under access structure $\mathcal{P}$, the owner fetches the corresponding registered public keys from the ledger, runs the encryption algorithm, and outputs a ciphertext. The ciphertext is therefore bound to the specific registered keys chosen at encryption time. Correctness guarantees that only a coalition of secret key holders whose registered keys satisfy $\mathcal{P}$ can reconstruct the masking value and hence recover M.
5. *Collaborative decryption.* A coalition of users controlling a satisfying set of registered secret keys computes local decryption shares for their respective ciphertext leaves and sends them to the combiner. Once sufficiently many valid leaf contributions are available to satisfy the access tree, the combiner applies the LSSS reconstruction procedure to recover the masking value and then decrypts the ciphertext. If the contributed keys do not satisfy the access structure, the reconstruction fails and no information about plaintext is revealed.

This decentralized design realizes *policy-governed coalition decryption over a public-key directory*, rather than conventional ABE semantics: ciphertexts are bound to access-structure (including public keys) selected at encryption time, and decryption is explicitly performed by a qualifying coalition.

3.2 Instantiation over Finite Cyclic Groups

We now give a concrete classical instantiation in a finite cyclic group under the discrete logarithm assumption. All security claims for this classical instantiation are proved in the random-oracle model under Assumption 1.

- **Setup**$(\lambda) \to \mathrm{PP}$
  1. Given λ, select a large prime p.
  2. Generate a cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p with generator g.
  3. Define a global universe of access labels $\mathcal{U}$.
  4. Output the public parameters $\mathrm{PP}_{\mathrm{pq}} = (p, \mathbb{G}, g, \mathcal{U})$.
- **KeyGen**$(a, \mathrm{PP}) \to (t_a, T_a)$
  1. Sample $t_a \xleftarrow{\$} \mathbb{Z}_p$.
  2. Compute $T_a \leftarrow g^{\,t_a} \in \mathbb{G}$.
  3. **Output:** (t_a, T_a).

The public registration (a, T_a), together with a digital signature and any ledger metadata (e.g. identity, timestamp, registration identifier), is then published to the immutable ledger. For a fixed ciphertext, each leaf of the access tree refers to one such concrete registered public key.

Access-Structure Representation: Access tree syntax (following [10] only at the level of tree notation). An access structure is a rooted tree T. Each internal node x carries a threshold pair (k_x, num_x) with $1 \le k_x \le \mathrm{num}_x$; each leaf node is bound to a concrete registered public key, which we denote by its ledger label a, and has $k_x = 1$. The children of every node are ordered $1, \ldots, \mathrm{num}_x$.

Satisfaction: Let Γ denote the set of registered leaf keys whose secret counterparts are actually contributed in a decryption attempt. We define recursively

$$T_x(\Gamma) = \begin{cases} 1 & \text{if } x \text{ is a leaf and } \mathsf{reg}(x) \in \Gamma, \\ 1 & \text{if } x \text{ is internal and at least } k_x \text{ children } x' \text{ satisfy } T_{x'}(\Gamma) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

We write $\Gamma \models T$ when $T_{\mathrm{root}}(\Gamma) = 1$.

Encryption Algorithm: encrypts a message under an access structure whose leaves are bound to registered public keys.

- **Encrypt**$(PP, m, T) \to \mathsf{CT}$
  1. Fetch T_a for each $a \in \mathrm{leaves}(T)$.
  2. Sample $s \xleftarrow{\$} \mathbb{Z}_p$ and set $C_0 \leftarrow m \cdot g^s$.
  3. Run SHARE$(\mathrm{root}(T), s)$, where:
     (a) If v is a leaf bound to a, set $C_a \leftarrow T_a^{\rho} = g^{t_a \rho}$.
     (b) Else if v is a threshold node (τ, n), sample $q_v \in \mathbb{Z}_p[x]$ of degree $\tau - 1$ with $q_v(0) = \rho$, and recurse on each child v_j using $q_v(j)$.
  4. **Output:** $\mathsf{CT} = (C_0, \{C_a\}_{a \in \mathrm{leaves}(T)})$.
- **Decrypt**$(PP, \mathsf{CT}, SK_S, T) \to m \in \mathbb{G}$ or $\perp$

1. For each qualifying $a \in S$, compute $D_a \leftarrow C_a^{t_a^{-1}} = g^{\rho_a}$.
2. For each satisfied threshold node, choose τ valid child shares $S_\tau = \{(y_i, D_i)\}_{i=1}^{\tau}$, compute $\lambda_i = \prod_{j \neq i} \frac{y_j}{y_j - y_i}$, and reconstruct $D_{\text{parent}} \leftarrow \prod_{i \in S_\tau} D_i^{\lambda_i}$. Repeating upward yields $D_{\text{root}} = g^s$.
3. **Output:** $\left(C_0 \cdot (D_{\text{root}})^{-1}\right)$ if successful, else $\perp$.

The algorithm succeeds if and only if the contributed registered keys form a satisfying coalition for the access structure. Thus, the construction realizes access-structured, policy-governed *coalition decryption* over independently generated keys, under Assumption 1.

Decryption Correctness

Theorem 1 (Decryption Correctness). *Let T be an access tree whose leaves are bound to ledger-registered public keys $\{T_a = g^{t_a}\}_{a \in \text{leaves}(T)}$. Let $\mathsf{CT} = (C_0, \{C_a\}_{a \in \text{leaves}(T)})$ be an encryption of a message $m \in \mathbb{G}$. Suppose a coalition of users contributes the secret keys corresponding to a set of registered leaves $\Gamma \subseteq \text{leaves}(T)$ such that $\Gamma \models T$. Then the decryption algorithm recovers plaintext m*

Proof. The proof described in Appendix B

3.3 Post–Quantum Instantiation over RLWE

We now present an RLWE-based instantiation of DASTE that is intended to remain secure against quantum adversaries. As in the classical instantiation, the goal is to encrypt to an *access structure over concrete registered public keys*, so that only a qualifying coalition can reconstruct the masking term and recover the message.

– **DASTE.Setup**(λ):
1. Given λ, select the ring dimension N, ciphertext modulus $q > 2$, polynomial ring $R_q = \mathbb{Z}_q[x]/(x^N + 1)$, error distribution χ, plaintext modulus p, plaintext ring $R_p = \mathbb{Z}_p[x]/(x^N + 1)$, gadget parameters (w, ℓ) with $\ell = \lceil \log_w q \rceil$, and a uniformly sampled global mask polynomial $\alpha \xleftarrow{\$} R_q$.
2. Output the public parameters $\text{PP}_{\text{pq}} = (N, q, \chi, p, (w, \ell), \alpha)$

The public parameter PP_{pq} is published on the ledger. No master secret is generated.

User Key Generation and Registration. Each user independently generates key material for every access label a that it wishes to register.

– **DASTE.KeyGen**(PP_{pq}, a):
1. Sample $t_a \xleftarrow{\$} \chi$ such that t_a is invertible in R_q.
2. Compute the private decryption key $v_a \leftarrow t_a^{-1} \bmod q$.
3. For each $l = 0, 1, \ldots, \ell$,
   (a) Sample noise

$e_{a,l} \xleftarrow{\$} \chi$.

4. Compute
$$T_{a,l} \leftarrow \alpha \cdot (w^l \cdot t_a) + p \cdot e_{a,l} \pmod q.$$
5. Set $T_a = \{T_{a,l}\}_{l=0}^{\ell}$.
6. *Output:* the registered public key T_a and secret key v_a.

The private key $v_a = t_a^{-1} \bmod q$ is retained solely by the user. Thus, key generation is fully decentralized: every registered leaf key used in encryption is generated independently by its owner, and no central authority learns any decryption secret.

Publishing the public key to the ledger. The user signs using a post-quantum signature scheme, such as ML-DSA / Dilithium [16], and publishes the signed registration in the immutable ledger. Concretely, the user:

- generates a signature key pair $(sk_{\text{sig}}, pk_{\text{sig}})$;
- computes a signature
$$\sigma_a = \text{Sign}(sk_{\text{sig}}, (a, T_a, \mathsf{timestamp}));$$
- publishes the tuple (a, T_a, σ_a) to the ledger.

The ledger therefore acts as a public directory of registered decryption keys. A ciphertext does not target an abstract attribute in the ABE sense; rather, each leaf of the access tree is bound to a *concrete registered key instance* selected from this ledger.

Access-Structure Representation.

Exactly as in the classical instantiation (Section 3.2), the data owner specifies an access tree T whose internal nodes are threshold gates (τ, n). Each leaf is associated with a concrete ledger registration a, i.e. a specific public key T_a fetched from the public directory.

Encryption

Encryption proceeds by sampling a global masking secret s, sharing it over the access tree, and binding each resulting leaf share to the corresponding registered public key.

GenShare(R_q, v, ρ)

1. Initialize $L \leftarrow \varnothing$.
2. Define $\textsc{Share}(R_q, v, \rho)$ recursively:
   (a) **if** v is a leaf bound to a, set $L \leftarrow L \cup \{(a, s_a \leftarrow \rho)\}$.
   (b) **else if** v is a threshold node (τ, n), compute $\{\rho_j\}_{j=1}^{n} \leftarrow \text{ESSS.Split}(R_q, n, \tau, \rho)$, and call $\textsc{Share}(R_q, v_j, \rho_j)$ for each child v_j.
3. *Output:* L.

DASTE.Encrypt$(m, \text{PP}_{\text{pq}}, T, \{T_a\}_{a \in \text{leaves}(T)})$

1. Sample $s, e \xleftarrow{\$} \chi$.
2. Set $C_0 \leftarrow \alpha s + pe + m \pmod q$.
3. Compute $\{(a, s_a)\}_{a \in \text{leaves}(T)} \leftarrow \textsc{GenShare}(R_q, \text{root}(T), s)$.
4. For each $a \in \text{leaves}(T)$ and $l = 0, \ldots, \ell$, sample $e'_{a,l} \xleftarrow{\$} \chi$ and set $C_{a,l} \leftarrow s_a T_{a,l} + pe'_{a,l} \pmod q$.
5. Erase $\{s_a\}$.
6. *Output:* $\mathsf{CT} = (C_0, \{C_a\}_{a \in \text{leaves}(T)})$.

The resulting ciphertext is bound to the concrete set of registered public keys selected at encryption time. Hence, newly registered users cannot retroactively decrypt old ciphertexts unless their keys already appear as leaves in the ciphertext's access tree.

Decryption. Suppose a coalition of users contributes secret keys $\{v_a\}_{a\in S}$ whose corresponding registered leaves satisfy the access tree T. Decryption proceeds in two stages: local share opening and global reconstruction.

GenLocalShare$(\mathsf{CT}, v_a, a, \mathrm{PP}_{\mathrm{pq}})$

1. Sample $e'' \leftarrow \chi$.
2. Compute $(v_{a,0}, \ldots, v_{a,\ell}) \leftarrow \mathrm{BitDecompose}_{w,\ell}(v_a)$.
3. Set $D_a \leftarrow \sum_{l=0}^{\ell} v_{a,l} C_{a,l} + pe''$.
4. *Output:* D_a.

GenGlobalKey$(\mathsf{CT}, T, \{D_a\}_{a\in S}, \mathrm{PP}_{\mathrm{pq}})$

1. Let $\{\lambda_a\}_{a\in S}$ be the LSSS interpolation coefficients for the satisfying coalition S.
2. Set $D \leftarrow \sum_{a\in S} \lambda_a D_a$.
3. *Output:* D.

Decrypt(CT, D)

1. Set $m' \leftarrow C_0 - D \pmod q \pmod p$.
2. *Output:* m'.

Thus, decryption succeeds only when a qualifying coalition contributes enough registered secret keys to satisfy the access structure. Each participant opens only its own leaf component, and the final masking term is recovered through LSSS interpolation across the coalition. This realizes post-quantum, access-structured coalition decryption over independently generated, ledger-registered keys.

3.4 Correctness of Decryption

We now show that the post-quantum DASTE decryption procedure correctly recovers the plaintext whenever a qualifying coalition contributes registered secret keys whose corresponding leaves satisfy the access tree.

Theorem 2 (Decryption Correctness). *Let T be an access tree whose leaves are bound to registered public keys $\{T_a\}_{a\in\mathrm{leaves}(T)}$, and let $\mathsf{CT} = (C_0, \{C_a\}_{a\in\mathrm{leaves}(T)})$ be a ciphertext produced by **DASTE.Encrypt**. Suppose a coalition $S \subseteq \mathrm{leaves}(T)$ satisfies the access structure. For each $a \in S$, let D_a be the local decryption share output by **GenLocalShare**. Assume that the effective local noise terms ν_a satisfy $\nu_a \in pR_q$ for all $a \in S$. Then the decryption algorithm recovers the original message: $m' = m$.*

Proof. The proof is described in Appendix A

3.5 Ciphertext-Side Access-Structure Evolution

In the post-quantum DASTE instantiation, registered decryption keys $\{v_a = t_a^{-1} \bmod q\}_a$ are immutable: users generate them once, publish the corresponding

public registrations $T_a = \{T_{a,l}\}_{l=0}^{\ell}$, and retain the private keys locally. Accordingly, any post-encryption change to the access structure must be realized *solely by updating the ciphertext*, without modifying user keys or ledger registrations.

Let T be the current access tree. For each internal node x, let q_x denote the local sharing polynomial of degree $k_x - 1$, where k_x is the threshold at x and $q_x(0) = s_x$ is the share held at x. For each leaf a, the corresponding share $s_a = q_{\mathsf{parent}(a)}(\mathrm{index}(a))$ is stored only implicitly through the ciphertext components

$$C_{a,l} = s_a T_{a,l} + pe'_{a,l} \pmod q, \qquad l = 0, \ldots, \ell.$$

After each update, ciphertext components are recomputed from the updated shares with fresh noise, and all explicit shares are erased.

We consider three public transformations on the access tree: insertion of a semantically neutral gate, escalation of a threshold requirement, and revocation of a subtree.

Ciphertext Re-randomization. After any structural edit, the data owner may refresh the ciphertext under the updated tree. Sample a fresh mask $s^\star \xleftarrow{\$} \chi$ and rerun the sharing procedure on the current tree to obtain fresh leaf shares $\{s_a^\star\}_{a \in \mathrm{leaves}(T)}$. For each leaf a, replace its component by

$$C_a^\star = \left\{C_{a,l}^\star = s_a^\star T_{a,l} + pe_{a,l}^\star \pmod q\right\}_{l=0}^{\ell}, \qquad e_{a,l}^\star \xleftarrow{\$} \chi.$$

This preserves decryptability under the updated access structure while producing a ciphertext distributed as a fresh encryption.

T1: Semantically Neutral Gate Insertion Let x be a node in the access tree, and insert a new parent y above x, where y is a $(1,1)$-gate. Define q_y by

$$q_y(0) = q_{\mathsf{parent}(x)}(\mathrm{index}(y)), \qquad q_y(\mathrm{index}(x)) = q_x(0).$$

Because y has threshold 1 and only one child, the share passed to x is unchanged. Hence no leaf share changes and no ciphertext component must be updated:

$$C'_a = C_a \qquad \text{for all } a \in \mathrm{leaves}(T).$$

This operation is therefore purely structural.

T2: Threshold Escalation Let x be an internal node currently enforcing a (t,n)-threshold rule. We consider two ways of increasing the threshold requirement by one.

Case T2.1: $(t, n) \to (t+1, n)$. The child set remains unchanged, but the threshold at x is raised from t to $t+1$. Replace the sharing polynomial by $q'_x(X) = (X+1)\, q_x(X)$. For each child y of x, define $d_y = \mathrm{index}(y) + 1$. Then every ciphertext component in the subtree rooted at y is rescaled as

$$C'_{a,l} = d_y\, C_{a,l} + pe^{\mathrm{new}}_{a,l} \pmod q, \qquad e^{\mathrm{new}}_{a,l} \xleftarrow{\$} \chi,$$

for all leaves a beneath y and all $l = 0, \ldots, \ell$. Equivalently,

$$C'_a = \left\{ d_y\, C_{a,l} + pe^{\mathrm{new}}_{a,l} \pmod q \right\}_{l=0}^{\ell}.$$

Case T2.2: $(t, n) \to (t+1, n+1)$. A new child subtree T_z is inserted beneath x, and the threshold is simultaneously raised from t to $t+1$. If the new child is inserted at local index v, define

$$a = -\frac{1}{v}, \qquad q'_x(X) = (aX+1)\, q_x(X).$$

For each pre-existing child y, set $d_y = a \cdot \mathrm{index}(y) + 1$ and update every ciphertext component in the subtree rooted at y by

$$C'_{a,l} = d_y\, C_{a,l} + pe^{\mathrm{new}}_{a,l} \pmod q, \qquad e^{\mathrm{new}}_{a,l} \xleftarrow{\$} \chi.$$

For the new subtree T_z, the induced root share is zero: $q_z(0) = 0$. Hence each newly introduced leaf $b \in \mathrm{leaves}(T_z)$ is initialized as

$$C_{b,l} = pe_{b,l} \pmod q, \qquad e_{b,l} \xleftarrow{\$} \chi, \qquad l = 0, \ldots, \ell.$$

Ciphertext rerandomization may then be applied to refresh the full ciphertext under the updated tree.

T3: Subtree Revocation Let x be an internal node with threshold rule (t, n), and let y be one of its child subtrees. Revoking T_y removes that subtree from the authorization structure, replacing the parent rule (t, n) by $(t, n-1)$, provided $t \le n - 1$.

Operationally, the update deletes all ciphertext components associated with leaves in the revoked subtree: $\{C_a\}_{a \in \mathrm{leaves}(T_y)}$. All remaining components are then refreshed by rerandomization on the updated tree. As a result, any coalition whose authorization depended on T_y is no longer able to decrypt, while all registered secret keys remain unchanged.

These transformations provide a ciphertext-only interface for maintaining access policies after encryption. They preserve decentralized key ownership, require no user interaction, and allow the authorization structure to evolve entirely at the ciphertext layer.

4 Security Analysis

DASTE is analyzed under a *coalition-based* confidentiality model. Unlike collusion-resistant ABE, a satisfying coalition is not considered an attack: it is the intended decryptor. Accordingly, security is required only against adversaries that do *not* control a set of registered leaf keys satisfying the challenge access structure.

4.1 Security Model

The ledger is public and append-only. Hence the adversary may observe all public parameters, all registered public keys, all policy commitments, and all ciphertexts. It may also register arbitrary public keys of its own and corrupt any subset of registered users to obtain their secret keys.

For a challenge ciphertext encrypted under an access tree $T^\star$, let $L(T^\star)$ denote the set of registered leaves appearing in that ciphertext. A coalition $S \subseteq L(T^\star)$ is *authorized* if $S \models T^\star$. DASTE aims to provide IND-CPA security against every polynomial-time adversary whose corrupted challenge leaves form an *unauthorized* coalition, i.e.,

$$S^\star \not\models T^\star.$$

This captures the intended semantics of the scheme: ciphertexts are bound to concrete registered public keys selected at encryption time, and decryption is performed collaboratively by a qualifying coalition.

4.2 Coalition-Bounded IND-CPA Game

The confidentiality game is defined as follows.

1. **Setup.** The challenger runs $\mathsf{Setup}(1^\lambda)$, publishes the public parameters, and initializes the public ledger.
2. **Registration and corruption.** The adversary may adaptively register public keys and corrupt any users of its choice.
3. **Challenge.** The adversary submits two equal-length plaintexts m_0, m_1, a challenge access tree $T^\star$, and the registered public keys associated with the leaves of $T^\star$.
4. **Admissibility.** Let $S^\star$ be the set of challenge leaves whose secret keys are known to the adversary. The challenge is admissible only if

$$S^\star \not\models T^\star.$$

5. **Challenge ciphertext.** The challenger samples $b \xleftarrow{\$} \{0,1\}$, encrypts m_b under $T^\star$, and returns the resulting ciphertext $\mathsf{CT}^\star$.
6. **Guess.** The adversary outputs b', and its advantage is

$$\mathrm{Adv}_{\mathcal{A}}^{\text{ind-cpa}}(\lambda) = \left| \Pr[b' = b] - \frac{1}{2} \right|.$$

A DASTE instantiation is IND-CPA secure if this advantage is negligible for every polynomial-time adversary.

4.3 Classical Security

In the classical construction, the headline component is $C_0 = m \cdot g^s$, while each leaf component encrypts a secret share ρ_a of s under the registered public key $T_a = g^{t_a}$: $C_a = T_a^{\rho_a} = g^{t_a \rho_a}$. A holder of the matching secret exponent t_a can locally open this leaf to obtain $D_a = C_a^{t_a^{-1}} = g^{\rho_a}$. Only a satisfying coalition can reconstruct g^s via LSSS interpolation and remove the masking term from C_0.

Theorem 3 (Classical IND-CPA Security). *Under Assumption 1, the classical DASTE instantiation is IND-CPA secure against every probabilistic polynomial-time adversary that does not control a satisfying coalition of registered leaves in the challenge ciphertext.*

Proof. Under the discrete-log hardness assumption in the chosen group model, the challenge plaintext is hidden behind the mask g^s, and each leaf component reveals only an encryption of a share of s under the corresponding registered public key. An unauthorized coalition cannot reconstruct the root share and therefore cannot distinguish the challenge message except with negligible probability.

4.4 Post-Quantum Security

In the RLWE instantiation, the headline component is

$$C_0 = \alpha s + pe + m \pmod{q},$$

and each leaf component encrypts a share s_a of s under the corresponding registered public key:

$$C_{a,l} = s_a T_{a,l} + pe'_{a,l} \pmod{q}.$$

A user holding $v_a = t_a^{-1}$ locally computes a noisy opening $D_a \approx \alpha s_a$, and a satisfying coalition combines these local values using the LSSS coefficients to obtain $D \approx \alpha s$. Subtracting this from C_0 recovers the plaintext modulo p.

Theorem 4 (Post-Quantum IND-CPA Security). *Assuming decisional RLWE over R_q is hard, the post-quantum DASTE instantiation is IND-CPA secure against every probabilistic polynomial-time adversary that does not control a satisfying coalition of registered leaves in the challenge ciphertext.*

Proof. The proof follows a standard hybrid argument. Starting from the real challenge ciphertext, the reduction replaces the RLWE-derived public-key and ciphertext components associated with unreconstructed shares by uniformly random ring elements. By the decisional RLWE assumption, each replacement is computationally indistinguishable. Since the adversary's corrupted leaves do not satisfy $T^\star$, the corresponding LSSS shares are insufficient to reconstruct the root secret s. Thus the mask αs in C_0 remains hidden, and the challenge component is indistinguishable for m_0 and m_1. Any adversary distinguishing the challenge bit with non-negligible advantage would therefore yield a distinguisher for decisional RLWE, contradicting the assumption.

Thus the proof itself is unchanged; only the challenge restriction is interpreted under the coalition-based DASTE model rather than ABE semantics.

4.5 Security of Policy Evolution

The policy-evolution procedures of Section 3.5 act only on public information: the ciphertext, the public access tree, and the registered public keys. They do not access any user secret key.

Operation T1 (semantically neutral insertion) preserves the ciphertext semantics exactly. Operation T2 (threshold escalation) replaces the relevant local sharing polynomial and updates the public ciphertext components so that only coalitions satisfying the *updated* threshold can decrypt. Operation T3 (subtree revocation) removes all ciphertext components associated with the revoked subtree and refreshes the remaining components, thereby removing the revoked subtree from the authorized coalition set. Finally, rerandomization samples a fresh masking secret and fresh noise on the current tree, yielding a ciphertext distributed as a fresh encryption under the updated policy.

Hence policy evolution preserves the same confidentiality notion as the base scheme: after any public update, the resulting ciphertext remains IND-CPA secure against every coalition that does not satisfy the *updated* access structure.

4.6 Auditability and Trust Reduction

DASTE also provides two important system-level guarantees. First, there is no master secret and therefore no key-escrow authority: every user generates and retains its own decryption key locally. Second, because registered public keys and policy commitments are signed and recorded on an immutable ledger, the set of eligible decryptor keys for a ciphertext is publicly auditable. These properties are complementary to IND-CPA confidentiality and are particularly relevant in regulated, multi-tenant, and cross-organizational deployments.

5 Comparative Analysis

We position DASTE using four comparison dimensions that we view as fundamental from both a cryptographic and a trust-architecture perspective.

Authorization semantics. The first dimension is *who is intended to decrypt*. In conventional ABE, encryption is predicate-based and decryption is typically performed by a *single* user key satisfying the policy; collusion across users is therefore treated as a threat to be prevented [1, 10]. By contrast, DASTE is designed for *coalition authorization*: decryption is intentionally carried out by a set of registered key holders whose leaves satisfy the access structure. Under this dimension, DASTE is much closer to threshold encryption and ad-hoc threshold encryption than to CP-ABE.

Trust and key-generation model. The second dimension is *how decryption capabilities are created*. Early dynamic threshold encryption allows the sender to choose the recipient set and threshold at encryption time, but still relies on

system-side enrollment or join procedures [5]. Silent-setup threshold encryption removes this setup interaction by letting users generate keys locally [9]. Registered ABE also reduces trust in a central authority, but does so through a *key curator* that aggregates or manages registrations [11, 13]. DASTE adopts the strongest decentralization point in this spectrum: every user generates its own key material independently, and the ledger acts only as a public registration and audit layer. Thus, DASTE removes both PKG-style key escrow and curator dependence.

Policy binding model. The third dimension is *what the ciphertext is bound to.* In ABE, ciphertexts are associated with abstract attributes or attribute policies [1,10]. In public-key-directory encryption, such as flexible broadcast encryption, ciphertexts are bound to registered public keys in a directory [7]. DASTE follows the directory-based view, but strengthens it by binding the ciphertext to an *access structure over concrete registered keys.* Consequently, a ciphertext in DASTE is tied to a specific registration snapshot: newly registered users do not retroactively gain access unless their registered keys were already embedded as leaves in the ciphertext.

Post-encryption policy control. The fourth dimension is whether the access condition remains fixed once encryption is performed. Dynamic threshold encryption and silent-setup threshold encryption let the sender choose a threshold or recipient set at encryption time [5,9], but they do not natively provide a public ciphertext-update interface for later policy modification. Likewise, directory-based encryption focuses on public-key selection rather than ciphertext-side policy maintenance [7]. DASTE instead supports *ciphertext-side access-structure evolution*: semantically neutral insertion, threshold escalation, subtree revocation, and rerandomization can all be carried out without reissuing user secret keys. This makes DASTE especially suitable for environments where authorization requirements must evolve after deployment while preserving decentralized key ownership.

Taken together, these dimensions place DASTE most naturally in the space of *decentralized access-structured threshold encryption over registered public keys.* Its closest relatives are dynamic / silent-setup threshold encryption and public-key-directory encryption, rather than ABE. Relative to those lines, DASTE combines coalition-based authorization, independently generated registered keys, expressive access-tree policies, and ciphertext-only policy evolution in a single framework.

In summary, DASTE differs from conventional ABE, registered ABE, threshold encryption, and directory-based broadcast encryption along four main dimensions: authorization semantics, trust model, ciphertext binding, and post-encryption policy control. In CP-ABE and KP-ABE, decryption is typically performed by a single qualified user and collusion is treated as an attack [1,10]; registered ABE reduces authority trust but still relies on registration/curator mechanisms [11,13]. Dynamic and silent-setup TPKE support coalition-style decryption, but mainly for threshold policies and do not provide a native ciphertext-side

Table 1. Comparison with related access-control primitives.

Primitive	**Trusted Authority**	**Coalition**	**Policy**	**Post Quantum**	**Policy evolution**
CP/KP-ABE [1, 10]	Yes PKG/master secret	No	Expressive attributes	Usually no	No
Registered ABE [11, 13]	Partial key curator	No	Expressive attributes	No	Limited
Dynamic TPKE [5]	Yes/Partial enrollment	Yes	Threshold	No	No
Silent TPKE [9]	No silent setup	Yes	Threshold	No	No
Directory broadcast encryption [7]	No trusted dealer	No	Recipient directory	No	Limited
DASTE	No	Yes	Access tree/LSSS over registered keys	Yes, RLWE	Yes

policy evolution interface [5, 9]. Directory-based broadcast encryption binds ciphertexts to public-key directories [7], but focuses on recipient selection rather than expressive access-structure updates. In contrast, DASTE combines independently generated registered keys, coalition decryption, access-tree/LSSS policies over concrete key registrations, post-quantum RLWE security, and ciphertext-only policy evolution in one framework. Further details are presented in Table 1.

6 Conclusion

We presented *DASTE*, a decentralized, ledger-anchored encryption framework for coalition-based access control over independently registered public keys. Unlike conventional ABE, DASTE binds ciphertexts to concrete registered key instances selected at encryption time and treats collaborative decryption by a qualifying coalition as the intended functionality. Our construction supports expressive access structures, ciphertext-only policy evolution, and public auditability through immutable ledger registration, while avoiding trusted key-issuing authorities and key escrow.

We presented a classical discrete-log instantiation as a conceptual baseline and a post-quantum RLWE instantiation as the main construction, establishing coalition-bounded IND-CPA security for the RLWE variant under decisional RLWE. Taken together, these results show that decentralized key ownership, expressive threshold-style authorization, and post-encryption policy maintenance

can be achieved within a single cryptographic framework, making DASTE a promising primitive for auditable, adaptive, and post-quantum secure access control in distributed environments.

References

1. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2007)*, pages 321–334. IEEE Computer Society, 2007.
2. Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 97–106. IEEE, 2011.
3. Jeffrey Champion and David J. Wu. Distributed broadcast encryption from lattices. In *Theory of Cryptography (TCC 2024)*, volume 15366 of *Lecture Notes in Computer Science*, pages 156–189. Springer, Cham, 2025. Full version: IACR ePrint 2024/1417.
4. Vanesa Daza, Javier Herranz, Paz Morillo, and Carla Ràfols. Ad-hoc threshold broadcast encryption with shorter ciphertexts. *Electronic Notes in Theoretical Computer Science*, 192(2):3–15, May 2008.
5. Cécile Delerablée and David Pointcheval. Dynamic threshold public-key encryption. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 317–334. Springer, 2008.
6. Lalita Devadas, Abhishek Jain, Brent Waters, and David J. Wu. Succinct witness encryption for batch languages and applications. In *Advances in Cryptology – ASIACRYPT 2025*, volume 16252 of *Lecture Notes in Computer Science*, pages 130–162. Springer, Singapore, 2025.
7. Rachit Garg, George Lu, Brent Waters, and David J. Wu. Realizing flexible broadcast encryption: How to broadcast to a public-key directory. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security (CCS '23)*, pages 1093–1107. ACM, 2023.
8. Sanjam Garg, Mohammad Hajiabadi, Mohammad Mahmoody, and Ahmadreza Rahimi. Registration-based encryption: Removing private-key generator from IBE. In *Theory of Cryptography (TCC 2018)*, volume 11239 of *Lecture Notes in Computer Science*, pages 689–718. Springer, Cham, 2018.
9. Sanjam Garg, Dimitris Kolonelos, Guru-Vamsi Policharla, and Mingyuan Wang. Threshold encryption with silent setup. In *Advances in Cryptology – CRYPTO 2024*, volume 14926 of *Lecture Notes in Computer Science*, pages 352–386. Springer, Cham, 2024.
10. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pages 89–98. ACM, 2006.
11. Susan Hohenberger, George Lu, Brent Waters, and David J. Wu. Registered attribute-based encryption. In *Advances in Cryptology – EUROCRYPT 2023*, volume 14006 of *Lecture Notes in Computer Science*, pages 511–542. Springer, Cham, 2023.
12. Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. RFC 6962, June 2013.

13. George Lu, Brent Waters, and David J. Wu. Multi-authority registered attribute-based encryption. In *Advances in Cryptology – EUROCRYPT 2025*, volume 15603 of *Lecture Notes in Computer Science*, pages 3–33. Springer, Cham, 2025.
14. Marcela S. Melara, Aaron Blankstein, Joseph Bonneau, Edward W. Felten, and Michael J. Freedman. CONIKS: Bringing key transparency to end users. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 383–398, 2015.
15. National Institute of Standards and Technology. FIPS 203: Module-lattice-based key-encapsulation mechanism standard (ml-kem). Federal Information Processing Standards Publication, 2024.
16. National Institute of Standards and Technology. FIPS 204: Module-lattice-based digital signature standard (ml-dsa). Federal Information Processing Standards Publication, 2024.
17. C. Peikert. *A Decade of Lattice Cryptography*. 2015.
18. Brent Waters and David J. Wu. Silent threshold cryptography from pairings: Expressive policies in the plain model. IACR Cryptology ePrint Archive, Report 2025/1547, 2025. To appear in EUROCRYPT 2026.
19. Hoeteck Wee and David J. Wu. Unbounded distributed broadcast encryption and registered ABE from succinct LWE. In *Advances in Cryptology – CRYPTO 2025*, volume 16002 of *Lecture Notes in Computer Science*, pages 204–235. Springer, Cham, 2025.

A Proof of Theorem 2

For each $a \in S$, $T_{a,l} = \alpha(w^l t_a) + pe_{a,l} \pmod q$, so

$$C_{a,l} = s_a T_{a,l} + pe'_{a,l} = \alpha s_a (w^l t_a) + p(s_a e_{a,l} + e'_{a,l}).$$

Hence

$$D_a = \sum_{l=0}^{\ell} v_{a,l} C_{a,l} + pe''_a = \alpha s_a \sum_{l=0}^{\ell} v_{a,l} w^l t_a + \nu_a,$$

where

$$\nu_a = p\left(\sum_{l=0}^{\ell} v_{a,l}(s_a e_{a,l} + e'_{a,l}) + e''_a\right).$$

Since $\sum_{l=0}^{\ell} v_{a,l} w^l = v_a = t_a^{-1}$, it follows that $D_a = \alpha s_a + \nu_a$. If each $\nu_a \in pR_q$, then by linearity

$$D = \sum_{a\in S} \lambda_a D_a = \alpha \sum_{a\in S} \lambda_a s_a + \sum_{a\in S} \lambda_a \nu_a = \alpha s + \nu,$$

where $\nu \in pR_q$ because $\sum_{a\in S} \lambda_a s_a = s$. Therefore,

$$C_0 - D = (\alpha s + pe + m) - (\alpha s + \nu) = m + pe - \nu \pmod q.$$

Reducing modulo p removes both pe and ν, giving

$$m' = (C_0 - D) \bmod q \bmod p = m.$$

Thus any qualifying coalition decrypts correctly.

B Proof of Theorem 1

For each contributed leaf $a_i \in \Gamma$, local share opening yields

$$D_{a_i} = C_{a_i}^{t_{a_i}^{-1}} = (g^{t_{a_i}\rho_{a_i}})^{t_{a_i}^{-1}} = g^{\rho_{a_i}}.$$

Since $\Gamma \models T$, the recursive Shamir/LSSS reconstruction associated with T guarantees coefficients $\{\lambda_i\}_{a_i \in \Gamma}$ such that

$$\sum_{a_i \in \Gamma} \lambda_i \rho_{a_i} = s \pmod{p}.$$

Therefore,

$$\prod_{a_i \in \Gamma} D_{a_i}^{\lambda_i} = \prod_{a_i \in \Gamma} (g^{\rho_{a_i}})^{\lambda_i} = g^{\sum_{a_i \in \Gamma} \lambda_i \rho_{a_i}} = g^s.$$

Substituting this into the headline component $C_0 = m \cdot g^s$, we obtain

$$C_0 \cdot (g^s)^{-1} = m.$$

Hence any qualifying coalition whose registered leaves satisfy T correctly recovers the plaintext.

C Noise Analysis and Optimization using Gadget Technique

Noise-Bound Condition. In practice, parameters (ring dimension N, modulus q, noise distribution χ, and LSSS structure) are chosen so that

$$\|e\| + \|e''\| \ < \ q/2 \quad \text{and} \quad \|e - e''\| < p/2$$

after appropriate reductions, guaranteeing error-free recovery of m. The exact bound depends on the RLWE parameters and the size of $\sum_a \lambda_a v_a (s_a e_a + e'_a)$, which grows linearly in the noise terms and shares. Standard FHE-inspired parameter regimes ensure this bound holds for all intended policy depths and update operations.

This post-quantum instantiation therefore realises the same fine-grained, policy-driven access control as the classical scheme, while providing resilience against quantum adversaries.

Noise analysis. Let $T_a = \alpha t_a + p \cdot e_a$ with independent samples $t_a, e_a \leftarrow \chi$, and let $e'_a \leftarrow \chi$. The attribute component is therefore

$$C_a \ = \ s_a T_a + p \cdot e'_a \ = \ \alpha(s_a t_a) \ + \ p \cdot (s_a e_a \ + \ e'_a). \tag{4}$$

The first summand, $\alpha(s_a t_a)$, constitutes the signal, while the remaining terms represent noise. Because χ is zero-mean with variance σ^2, the product of two

independent samples has variance $\mathrm{Var}(s_a e_a) = N\sigma^4$, where N is the ring dimension. Consequently

$$\mathrm{Var}(C_a) \;\leq\; N\sigma^4 + \sigma^2, \tag{5}$$

exhibiting at most quadratic growth in the base noise parameter σ. This growth remains negligible relative to the modulus q under the standard RLWE parameter regime.