TU Delft

Delft University of Technology

Document Version
Final published version

Licence
CC BY

Citation (APA)
Darwish, M., Markatou, E. A., & Smaragdakis, G. (2026). A Policy-Based Conjunctive Scheme for Digital Forgetting of Co-Owned Data. *ACM Transactions on Privacy and Security*, *29*(3), Article 28. https://doi.org/10.1145/3811817

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

A Policy-Based Conjunctive Scheme for Digital Forgetting of Co-Owned Data

MARWAN DARWISH, cybersecurity, Delft University of Technology, Delft, Netherlands and cybersecurity, Radboud University, Nijmegen, Netherlands
EVANGELIA ANNA MARKATOU, Delft University of Technology, Delft, Netherlands
GEORGIOS SMARAGDAKIS, Delft University of Technology, Delft, Netherlands

In today's digital landscape, our interactions, from professional collaborations to personal data sharing involving photos, movies, and documents, have largely moved online. While transitioning these activities to digital platforms provides considerable convenience, it poses significant challenges in efficiently managing and securely erasing shared data in compliance with privacy regulations. Digital forgetting, particularly in co-owned data, transcends being merely desirable and becomes a mandate. Conventional data management paradigms, including cryptographic erasure techniques, typically apply uniform deletion across all stakeholders, neglecting audience-specific expiration and co-owner participation in deletion, which limits their applicability in contemporary cloud storage ecosystems. This article introduces a *Policy-Based Conjunctive Scheme* (PBCS) that enables conjunctive decision-making for data access and collaborative data forgetting, aligning with the *General Data Protection Regulation* (GDPR)'s *Right to be Forgotten* (RTBF). PBCS allows owners to upload their data to the cloud securely and offers policy-based access control to co-owners, granting them the ability to influence decisions about data deletion via democratic voting mechanisms significantly. The scheme leverages conjunctive access thresholds and mechanisms that gradually make data irretrievable. By integrating cryptographic primitives and Lagrange interpolation-based decay, PBCS supports a flexible, conjunctive governance model that upholds privacy and enhances the data lifecycle. We provide a formal analysis and an experimental evaluation of our scheme.

CCS Concepts: • **Security and privacy → Security services; Privacy-preserving protocols;**

Additional Key Words and Phrases: Co-Ownership Data Forgetting, Collaborative Data Governance, Policy-Based Access Control

ACM Reference Format:
Marwan Darwish, Evangelia Anna Markatou, and Georgios Smaragdakis. 2026. A Policy-Based Conjunctive Scheme for Digital Forgetting of Co-Owned Data. *ACM Trans. Priv. Sec.* 29, 3, Article 28 (May 2026), 37 pages. https://doi.org/10.1145/3811817

We thank our reviewers for their valuable feedback, which helped improve this article. This work was funded by the European Union under the Horizon Europe Programme as part of the projects RECITALS (Grant Agreement #101168490), SafeHorizon (Grant Agreement #101168562), and by the NWO Vidi project, A Web Security and Privacy Observatory (WeSPO). The content of this article does not reflect the official opinion of the European Union. The responsibility for the information and views expressed therein lies entirely with the authors.
Authors' Contact Information: Marwan Darwish (corresponding author), cybersecurity, Delft University of Technology, Delft, Netherlands and cybersecurity, Radboud University, Nijmegen, Netherlands; e-mail: marwan.darwishkhabbaz@ru.nl; Evangelia Anna Markatou, Delft University of Technology, Delft, Netherlands; e-mail: e.a.markatou@tudelft.nl; Georgios Smaragdakis, Delft University of Technology, Delft, Netherlands; e-mail: g.smaragdakis@tudelft.nl.



This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2026 Copyright held by the owner/author(s).
ACM 2471-2566/2026/05-ART28
https://doi.org/10.1145/3811817

1 Introduction

Cloud computing has revolutionized digital collaboration, significantly impacting how we share photos, videos, and documents. The global cloud storage market is projected to reach US$ 598.27 Billion by 2031, with a *Compound Annual Growth Rate* (CAGR) of 23.4% from 2023 to 2031 [69]. However, this evolution towards cloud storage services unveils data security vulnerabilities due to users' reduced control over their outsourced data [28, 41, 51, 55]. These issues raise critical concerns about the robustness and reliability of data deletion strategies [1, 44, 49, 64]. The enduring nature of digital data emerges as a double-edged sword. The Right to be Forgotten (RTBF), established by the EU General Data Protection Regulation (GDPR) [32, 34, 61, 75] in 2018, and the concept of ***Ephemerality*** (i.e., Temporary existence of data after a set period) [13, 16, 29] advocate for temporary data retention to enhance privacy. However, digital footprints resist deletion, remaining susceptible to replication, archival, and unauthorized sharing.

Today's Solutions. Concepts such as Vanish [36], EphPub [20], Ephemerizer [60], Safevanish [84], Timed Revocation [66] and Forgetting with Puzzles [5] use encryption, decentralized ephemeral storage such as *Distributed Hash Table* (DHT) [72] and *Domain Name System* (DNS) [52] to achieve digital data forgetting through key destruction during static or flexible expiration periods.

Challenges. Despite advancements, significant challenges remain in achieving audience-specific expiration (i.e., Multi-level forgetting [67]) and involving co-owners in deletion processes [2, 33, 35, 56]. Previous models, such as the Disjunctive forgetting scheme [27], introduced group-specific expiration keys via smart contracts but were limited by rigid user segmentation, which prevented overlapping group participation. Current approaches [5, 29, 36, 66, 81] lack co-owner involvement in access and deletion decisions, reducing flexibility in collaborative environments (i.e., multi-co-owner settings). Current systems are inadequate for the growth of cloud collaboration, highlighting the need for governance models that support dynamic group memberships, co-owner autonomy, and GDPR's RTBF.

Our approach. In response to these challenges, we present a ***Policy-Based Conjunctive Scheme (PBCS)***, a solution built on *Privacy by Design* principles [21] to address the limitations of previous models and to advance digital forgetting practices in cloud-based file-sharing. The scheme is well-suited to scenarios in which data owners and co-owners share responsibilities and rights regarding data access and deletion, such as in multi-author research projects. In these projects, participants collaborate on shared documents with varying access permissions. The project owner requires a system capable of dynamic copyright management, offering flexible access and deletion rights as the project evolves. Outdated drafts must be securely erased to uphold RTBF through collaborative governance as the project progresses. When providers fail to comply with deletion requests, PBCS utilizes novel key decay—a mechanism that gradually corrupts decryption keys over time—to ensure that data becomes irretrievable. This approach embodies a shift towards more democratic and adaptable data-forgetting frameworks [7, 42].

Our PBCS offers:

(a) *Flexible Access Control and Dynamic Revocation:* PBCS provides customized permissions for users with overlapping group memberships, effectively addressing varied collaboration needs. Moreover, the scheme facilitates dynamic revocation, allowing immediate invalidation of access when required.

(b) *Dynamic Ephemerality Data Control:* Key lifespans are adjusted based on specific audience requirements, aligning with evolving data retention policies.

(c) *Collaborative Data Forgetting Framework:* Integrating cryptographic techniques and policy-driven deletion, PBCS ensures data irretrievability and enables co-owner participation in deletion, supporting privacy and collaborative lifecycle management.

PBCS is designed to enhance the *collaborative management* of co-owned digital assets within cloud infrastructures, covering the *entire data lifecycle from inception to deliberate obsolescence*. In this context, a data owner is the primary individual or entity responsible for the data. At the same time, co-owners are additional stakeholders with shared rights and responsibilities over data access and deletion. The process begins with secure, encrypted uploads, where data owners use ephemeral keys that degrade over time, transitioning from complete integrity to predefined decay. This preserves data integrity for the necessary duration, reducing the risk of unauthorized access.

To accommodate overlapping group memberships, PBCS offers flexible user categorization and nuanced access controls across groups, enabling dynamic cross-group access—a key feature for collaborative environments. This reflects the scheme's conjunctive model, in which members must meet both collective and individual conditions for access and deletion. Ephemeral decryption keys are dynamically regenerated during the download phase based on validated conjunctive access, ensuring secure retrieval by individual or collaborative groups. Key expiration and access terms are governed by policies decided by the data owner. Co-owners can initiate data deletion through a governance framework that supports majority consensus (i.e., half or more participants) and vetoes voting rights (i.e., specific users) to prevent privilege escalation. This adaptability enables PBCS to efficiently respond to evolving data relevance and security needs. As data becomes less relevant, deletion schedules align with the keys' predefined lifespans, ensuring a smooth transition into the final stage, the "*forgetting phase*." During this phase, encryption keys degrade in accordance with the principles of RTBF, significantly reducing the risk of data breaches. Ephemeral keys ensure data inaccessibility over time, while co-owners initiate deletion via a semi-honest provider in accordance with policy. Any copies outside the protocol remain inaccessible without meeting the required conjunctive threshold, as key reconstruction depends on dynamic policies and decay. To finalize the lifecycle, co-owners verify deletion, ensuring data irrecoverability and lifecycle completion.

In summary, our study makes the following contributions:

- We introduce a cloud-based file-sharing framework for collaborative data management between owners and co-owners, ensuring data ephemerality in line with the GDPR's Right to be Forgotten.
- We provide a governance module that empowers co-owners to dictate content deletion, employing voting mechanisms.
- We develop access control mechanisms that enable group-specific permissions, flexible membership handling, and validation of data deletion in collaborative environments.
- We implement and experimentally evaluate PBCS on decay analysis, performance evaluation, and security.

2 Design Goals

2.1 Problem Statement

Can we design a collaborative digital management system that dynamically handles file sharing, ownership rights, and data forgetting while ensuring GDPR alignment (Article 17 and Recitals 65, 66)?

The rise of digital platforms, particularly in cloud storage, has exposed significant shortcomings in current data management frameworks, especially regarding co-owned data forgetting and complex user interactions [19, 67, 76]. Conventional systems are often inadequate for scenarios requiring nuanced data control and privacy measures [15, 31, 47, 65, 73, 74, 86]. In collaborative environments such as writing clubs, managing copyright terms dynamically across multiple authors poses complex

challenges. Existing systems must support precise, audience-specific data erasure to comply with RTBF. This challenge is magnified when multiple stakeholders have varying claims and rights over shared digital content [24, 54]. For example, members' contributions to a collective work in a writing club are subject to specific copyright terms that evolve over time [82]. The platform must dynamically adapt to these changes, ensuring old drafts can be forgotten-based audience or retained according to relevant conditions and legal agreements.

2.2 System Actors and Threat Models

The interaction of several actors is characterized in PBCS as follows:
***Data Owner* (DO):** Owner of the data objects. For each data object, the DO encrypts it and uploads it to the cloud. Additionally, the DO defines the policy for each data object, including access and expiration parameters that control the data lifecycle.
***Co-Owners* (COs):** The COs are authorized to access and delete co-owned data according to policies set by the DO.
***Cloud Provider* (CP):** The CP provides infrastructure for storing data, implementing the access control and deletion functionalities according to the policies defined by the DO.
The DO is considered fully trusted, ensuring secure and uninterrupted communication channels. The CP and COs are modeled as potential adversaries with the following behaviors:
Semi-Honest CP: The CP follows the protocol honestly (including enforcing access/deletion workflows and triggering policy-defined decay) due to legal and operational incentives, but is curious and attempts to extract unauthorized information from any information it observes, including stored data/metadata. If the CP deviates (e.g., refuses to delete or skips decay), it can compromise compliance/availability, while key confidentiality and post-expiry irrecoverability are preserved by the cryptographic design (explained in Section 4).
Malicious COs: COs might arbitrarily deviate from the protocol and engage in unauthorized actions, including **(i)** attempting to access data objects without proper authorization or exceeding their assigned permissions; **(ii)** initiating deletion requests for data without meeting the required voting thresholds or adhering to policy.

Furthermore, potential attack vectors are further discussed in Section 4.4.

3 Concepts

PBCS secures data d using a symmetric key k, not as a fixed secret but as a dynamically derived seed via *Key Sharing Scheme* (KSS). Unlike *Shamir Secret Sharing Scheme* (SSSS) [68], which splits a known secret into shares, ***KSS interpolates entropy-driven points into a Lagrange polynomial and extracts its free coefficient as the cryptographic seed for k. These transient points, Shares(d), evolve through passive entropy changes and active policy-driven decay***. They are encrypted with the co-owner's public keys and stored on the CP. A conjunctive threshold ensures only authorized COs can reconstruct k while it remains valid. Once enough shares degrade, k becomes irretrievable, enforcing cryptographic data forgetting.

3.1 Definitions

Table 1 lists the notations, with more definitions in Appendix A.

Definition 1. Our KSS extends SSSS by integrating entropy-based randomness for dynamic share generation. We define it:
ConstructKey: This randomized algorithm generates entropy-driven data points. Given a threshold θ and an entropy source S_{entropy}:

$$\text{KSS.ConstructKey}(\theta, S_{\text{entropy}}) \rightarrow (k, \{(x_i, y_i)\}_{i=0}^{\theta-1})$$

Table 1. Summary of Notations

Notation	Definition
θ	Threshold for reconstructing key k.
d	Data object.
τ	Time parameter in the decay function D.
S	External entropy source.
λ^*	Decay factor for key k.
(x_i, y_i)	Random data points.
SymEnc(d)	Symmetric encryption of d using a key k.
Share(d)	Data points (x_i, y_i) of key k.
AsymEnc(Shares(d))	Asymmetric encryption of key shares using public keys PKs.
$\text{Votes}_{\text{acc}}$	Voting shares for access control polynomial.
$\text{Votes}_{\text{del}}$	Voting shares for deletion polynomial.
policy(d)	Access and deletion policy for data object d.
$\text{Val}_{\text{initial}}$	Initial list of deletion validation.
m	Co-owners membership matrix.

*: λ is a key decay factor ensuring irretrievability, distinct from traditional cryptographic security parameters [12].

where S_{entropy} introduces external randomness from fluctuating real-world sources (e.g., cryptocurrency prices, stock trends, social media trends). These continuously changing values enhance unpredictability and enable passive decay.

ReconstructKey: The key is reconstructed from entropy-modified shares:

$$\text{KSS.ReconstructKey}(\{(x_i, y_i)\}_{i=0}^{\theta-1}) \rightarrow k$$

Definition 2. The decay function D ensures point degradation over time by combining entropy-driven randomness and policy-enforced control. Given $\tau \in \mathbb{R}^+$ (time), λ_{passive} (entropy-based decay), λ_{active} (owner-defined decay), and data points $(x_i, y_i) \in \mathbb{F}_q$: $D(\tau, \lambda_{\text{passive}}, \lambda_{\text{active}}, P_{\text{policy}}, (x_i, y_i)) \rightarrow (x_i', y_i')$, where x_i', y_i' evolve dynamically based on S_{entropy}, introducing randomness from external sources to ensure share unpredictability. If λ_{passive} decay is insufficient, P_{policy} enforces controlled deterioration through λ_{active}, supporting RTBF objectives. Decay guarantees key irretrievability when valid shares drop below the threshold θ.

```
{
  "Version": "YYYY-MM-DD",
  "Statement": [
    {
      "Effect": "Allow / Deny",
      "Action": [
        "Service:ActionType1",
        "Service:ActionType2"
      ],
      "Resource": "arn:service:region:account-id:resource",
```

```
        "Condition": {
          "ConditionType": {
            "aws:RequestTag/Role": "RoleName",
            "aws:RequestTag/Policy": "PolicyName"
          }
        }
      }
    ]
}
```

Listing 1. Generic JSON-Based IAM Policy

Definition 3. Policy P specifies permissions and conditions for data access based on roles and actions. The policy includes key elements such as version, allowed actions, effects, resources, and conditions. See Listing 1 for a sample policy format. Formally, the policy can be described as follows:

$$\text{Input} = \{\text{Statement}, \text{Effect}, \text{Action}, \text{Resource}, \text{Condition}\}$$

$$\text{Output} = \{\text{Policy for } \textit{Encrypted Data Object} \text{ (EDO)}\}$$

Definition 4. Our scheme $PBCS = (\text{Upload}, \text{Preprocessing}, \text{Access Control}, \text{Download}, \text{Deletion}, \text{Deletion Validation})$ consists of six algorithms:

Upload: Run by *DO*, this algorithm takes as input the data object ($d \in \{0,1\}^*$), the groups' public keys *PKs*, and outputs encrypted data using symmetric and asymmetric encryption for data points Shares(d), with $policy(d)$. The output is transmitted to the *CP*.

$$\text{Output} = \{\text{SymEnc}(d), \text{AsymEnc}(\text{Shares}(d)), policy(d)\}$$

Preprocessing: Run by the *CP*, this algorithm takes as input the group metadata, uploaded encrypted data, and policies, processing them to generate voting shares for deletion and access control. Furthermore, an initial list of intersection points between the original and deletion polynomials is generated for data deletion validation.

$$\text{Input} = \{\text{SymEnc}(d), \text{AsymEnc}(\text{Shares}(d)), policy(d)\}$$

$$\text{Output} = \{\text{Votes}_{\text{del}}, \text{Votes}_{\text{acc}}, \text{Val}_{\text{initial}}\}$$

Access Control: This algorithm, run by the *CP*, takes as input $\text{Votes}_{\text{acc}}$ and outputs an access decision (0 or 1).

$$\text{Input} = \{\text{Votes}_{\text{acc}}\}, \text{Output} = \{\text{Access decision}\}$$

Download: *COs* decrypt and retrieve d using the provided encrypted Share(d), keys, membership matrix, and threshold.

$$\text{Input} = \{\text{SymEnc}_{\text{Shares}(d)}(d), \text{AsymEnc}(\text{Shares}(d)), m, \theta\},$$

$$\text{Output} = d$$

Deletion: Initiated by *COs* and supported by *CP*, this algorithm takes as input $\text{Votes}_{\text{del}}$ and outputs a deletion decision (0 or 1).

$$\text{Input} = \{\text{Votes}_{\text{del}}\}, \text{Output} = \{\text{Deletion decision}\}$$

Deletion Validation: Run by *COs*, this algorithm verifies the final deletion status of data ($\text{Val}_{\text{initial}}$) via shared votes, outputting 0 or 1.

$$\text{Input} = \{\text{Shares}(d), \text{Votes}_{\text{del}}\}, \text{Output} = \{\text{Result verification}\}$$

3.2 Security Definitions

Correctness of Protocol Correctness in the PBCS protocol ensures that cryptographic operations align with Definitions 4 and 3. The protocol guarantees that downloaded data matches the original

upload via consistent key shares (Definition 1), while access control and deletion strictly follow policy rules.

Definition 5. A scheme $PBCS$ = (Upload, Preprocessing, AccessControl, Download, Deletion, Deletion Validation) is correct if the conditions hold:

(1) The d retrieved through the download matches the original upload: Download(Upload(d)[: 2], m, θ) = d.
(2) Only authorized COs are able to access or delete the data based on valid voting shares V: AccessControl/Deletion(V) = b, where $b = 1$ if the V comply with the policy(d), otherwise $b = 0$.
(3) Given d with decay rate $\lambda \in \mathbb{R}^+$ and reconstruction threshold $\theta \in \mathbb{N}$, at time $\tau \in \mathbb{R}^+$ the decay function $f(\tau, \lambda)$ renders fewer than θ points/shares valid for interpolation, i.e., $|\{i \mid (x_i(\tau), y_i(\tau)) \text{ is valid}\}| < \theta$, implying that collective co-owner shares are no longer sufficient to reconstruct d after τ.
(4) Validation outputs 0 if COs retrieve data; otherwise, it outputs 1.

Privacy of Inputs We adopt the system actors and threat model from §2.2. The protocol execution reveals only what the ideal model permits: the data size and encrypted coefficients to a semi-honest CP, and known access-control or deletion details. The policy restricts malicious COs from accessing or deleting d.

Definition 6. Let the leakage profile $\Lambda = (L_S, L_Q)$, where $L_S(d)$ represents the size of the data object d and L_Q describes the number of polynomial coefficients encrypted via asymmetric encryption in the protocol $PBCS$ = (Upload, Access Control,
Download, Deletion, DeletionValidation). The protocol is Λ-secure if there exists a PPT simulator SIM, such that for any adversary A interacting with a semi-honest CP and malicious COs, the adversary cannot distinguish between:

— The real execution, including encrypted object d, polynomial coefficients, access/deletion votes, and $Deletion_{validation}$.
— The simulated execution, constructed by SIM based on Λ, without access to actual d.

For all $x_i \in \mathbb{F}_p$, the probability that adversary A distinguishes between real and simulated views is negligible: $\Pr[A(V_{\text{real}}) \neq A(V_{\text{sim}})] \leq \epsilon$, where ϵ is negligible in the security parameter. Thus, $PBCS$ is Λ-secure.

3.3 Terminology

Conjunctive Scheme: This scheme allows data sharing among multiple co-owners across distinct groups, with access determined by individual or collective permissions by predefined policies. Leveraging a cryptographic multi-group approach enhances data management flexibility and complexity in cloud environments [6, 19, 78]. The scheme dynamically applies expiration and access control policies that support overlapping memberships, facilitating data forgetting across diverse groups.

Lagrange-Basis Polynomials: Lagrange polynomials [30, 63] enable interpolation to form a polynomial through specified points. Given n share/point pairs $\{(x_j, y_j)\}_{j=0}^{n-1} \subseteq \mathbb{F}_q^2$, they are formulated as $L(x) = \sum_{j=0}^{n-1} y_j \ell_j(x)$ with $\ell_j(x) = \prod_{\substack{0 \leq m \leq n-1 \\ m \neq j}} \frac{x - x_m}{x_j - x_m}$, facilitating key reconstruction from distributed shares.

Ephemerality Via Key Decay: This scheme uses Lagrange polynomial attributes to generate ephemeral keys that degrade over time (i.e., Conjunctive decay, detailed in Section 4.1.1). As the shares change, the reconstructed key's effectiveness reduces below a threshold, rendering it irrecoverable and supporting digital forgetting. Key decay is given by Decay $= 1 - \lambda$, dictating the deterioration rate.

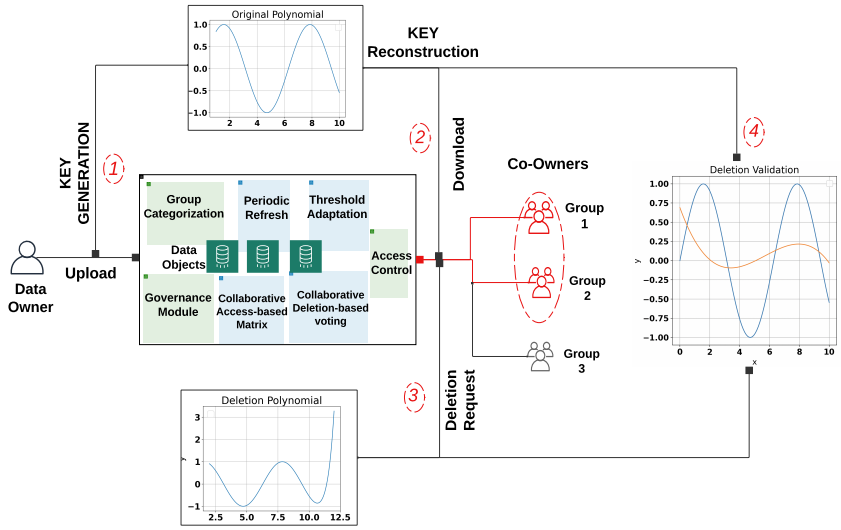


Fig. 1. Comprehensive architecture of the PBCS.

Access Control and Group Management: This module enforces access control using public-key cryptography and secret sharing by the CP. Group affiliations and permissions are validated through cryptographic signatures, where each signature binds a group identifier with a nonce for freshness. This ensures compliance with predefined policies and prevents unauthorized requests.
Governance Module: This module enables collaborative data deletion through majority or veto-based voting mechanisms. Managed by the CP in accordance with the owner's policies, it ensures flexible, responsive deletion decisions that adapt to user needs. We assume the Governance Module/CP is *semi-honest* for protocol execution (i.e., it follows prescribed policy enforcement, share/vote collection, and triggering of access/deletion/decay), while malicious CP behaviors (e.g., skipping computations, snapshot retention, or collusion with a subset of COs) are analyzed in Section 4.4. Threats outside the cryptographic scope include policy manipulation/tampering, inaccurate governance inputs (e.g., group membership or decay/entropy signals), identity compromise/Sybil behavior [84], selective message suppression, or a Byzantine CP that forges votes or tampers with deletion records; these require operational safeguards (e.g., COs-signed votes with an append-only audit log, or trusted execution) beyond our core cryptographic guarantees.

3.4 System Architecture

As outlined in Definition 4 and illustrated in Figure 1, our cryptographic framework collaboratively manages the data lifecycle between owners and co-owners in a cloud environment.

In the uploading phase—**Step (1)**, data owners encrypt and upload data (i.e., EDO) to the cloud using symmetric encryption. This process incorporates seeds derived from Lagrange polynomials used for symmetric-key generation, based on uniformly random inputs specified by the owners. Keys degrade as $D(\tau, \lambda_{\text{passive}}, \lambda_{\text{active}})$ modifies shares, with passive decay autonomous driven by entropy sources and active decay enforcing policy-controlled corruption. The key is irretrievable once shares drop below the threshold. Moreover, the co-owner's data points (i.e., used in symmetric key generation) are encrypted using asymmetric encryption, allowing only authorized co-owners to reconstruct the key during the download phase and preventing the provider from learning it. Users are categorized into groups established by the owner, dictating a precise policy framework

via shares tied to Lagrange polynomial points. The scheme dynamically adapts to group dynamics through flexible policies that accommodate real-time threshold changes (e.g., the minimum valid shares for a key) and periodic key refreshes to integrate new data points.

In the downloading phase—**Step** (2), co-owners verify access permissions through ID and signature validation (defined in Section 3.3) before decrypting data using a symmetric key. Each group employs its private key to decrypt its encrypted shares, collaboratively reconstructing the symmetric key k to enable data recovery. This phase accommodates collaborative (groups in red with overlapping co-owners) and individual access scenarios by generating combined or separate polynomials for decryption. Overlapping group memberships determine access flexibility, managed through a matrix-based method that identifies intersecting group memberships, facilitating a tailored polynomial reconstruction approach. This method dynamically generates polynomials for tailored decryption, enabling flexibility in access control.

In the deletion phase—**Step** (3), the governance module processes deletion requests with majority or veto voting (defined in Section 3.3). These approaches enable systematic data erasure through polynomial reconstruction before the key decays, rendering the data inaccessible. The deletion polynomial is derived from the collected deletion votes following a deletion request.

In the final phase—**Step** (4), the deletion validation mechanism analyzes overlaps between encryption and deletion polynomials. Constant intersections indicate continued accessibility, while variations signal sufficient decay of the decryption key, ensuring the content becomes inaccessible.

Art. 17 GDPR: RTBF (Recitals 65, 66) alignment is achieved through the system ensuring data irretrievability via CP- led deletion and autonomous key decay, safeguarding privacy rights. This mechanism focuses on private verifiability between DO and COs in the collaborative framework.

4 Scheme Description

We focus on (*i*) Background Analysis, (*ii*) Core Components, (*iii*) Formal Analysis, and (*iv*) Adversarial Models Analysis.

4.1 Background Analysis

4.1.1 Conjunctive Key Decay Scheme. The proposed PBCS introduces a hybrid key-decay mechanism that combines entropy-driven passive decay with policy-defined active decay tailored to audience-specific expiration needs. This eliminates reliance on ephemeral storage and prevents long-term archival retention.

Passive Decay and Entropy-Driven Evolution Key shares $K_i(t)$ dynamically evolve using multiple public time-varying sources (e.g., cryptocurrency prices, stock trends) to avoid static shares and hinder snapshot-based retention. We do not assume these sources provide cryptographic-quality entropy; they may be partially predictable or adversarially influenced. Using multiple independent sources reduces reliance on any single input and makes full source control harder in practice (and more resource-intensive). Instead, each access recomputes and hashes the updated coordinates, $x_i' = H(D(x_i, \lambda_{\text{passive}}, S_{\text{entropy}}, \tau))$ and $y_i' = H(D(y_i, \lambda_{\text{passive}}, S_{\text{entropy}}, \tau))$, so stored snapshots become stale, while key recovery still requires at least θ valid points under the threshold scheme (cf. [29]). To stabilize source variability, decay progression is controlled by averaging per-share inputs. The entropy-weighted decay is computed as: $\mu_w = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \cdot (K_x(t_i) + K_y(t_i))$, where $K_x(t_i)$ and $K_y(t_i)$ are the time-varying inputs for each share's coordinates. If passive evolution is insufficient or contested, the policy triggers deterministic active decay, and the system can refresh/re-key by reissuing new points under a new entropy configuration, invalidating prior snapshots (See Appendix B).

ALGORITHM 1: Upload

Input: Data Object d, Group public keys PKs
Output: SymEnc(d), AsymEnc($Shares(d)$), $policy(d)$

1 Generate $policy(d)$ and extract Threshold θ
2 Generate Key k, $data_points \leftarrow KSS.ConstructKey(\theta)$
3 **return** SymEnc(d,k), $\{\text{AsymEnc}(k_{pub}, \text{data_points})\}_{k_{pub} \in PKs}$, policy$(d)$

Active Decay and Policy-Defined Expiry When passive entropy changes are insufficient, policy-driven active decay enforces policy-defined expiration by modifying Lagrange coefficients. The policy dynamically perturbs selected coefficients embedded until a sufficient number of corrupted shares renders the key irrecoverable: $(x_i', y_i') = D(x_i, y_i, \lambda_{\text{active}}, P_{\text{policy}})$, where λ_{active} ensures controlled key expiration based on predefined policies by the owners.

Thresholds and policy mapping. In PBCS, θ denotes the *KSS key-reconstruction threshold*: download succeeds only while at least θ valid (non-decayed) points/shares remain, so θ together with the policy's passive/active decay parameters determines the key's validity window specified in $policy(d)$ by the DO. The DO defines these access/expiry thresholds in $policy(d)$ (e.g., via per-group thresholds θ_i under conjunctive access and/or a collaborative threshold θ_{collab}), and the upload/download procedures extract and apply them when constructing and reconstructing k (see Section 4.2.3). Concretely, letting n denote the number of policy-relevant groups/clauses for the object at time t, decryption is permitted only if:

$$I_P(t) = \begin{cases} 1, & \text{if } \forall i \in \{1, \dots, n\} \ : \ K_i(t) \geq \theta_i \ \text{ or } \ K(t) \geq \theta_{\text{collab}}, \\ 0, & \text{otherwise,} \end{cases}$$

and key irretrievability occurs once $K(t) < \theta_{\text{collab}}$. Deletion is authorized independently via the policy's deletion rule (e.g., majority threshold T_m, veto condition, and deletion activation Δ in the deletion polynomial), and does not reuse θ.

4.2 A Policy-Based Conjunctive Scheme (PBCS)

The PBCS framework enables dynamic file sharing and secure data forgetting in multi-owner cloud environments, adhering to GDPR Recitals 65 and 66, and Article 17 [32, 34, 61, 75]. It manages secure data transitions across the upload, download, delete, and forget phases (Definition 4). Additional algorithms are presented in Appendix D.

4.2.1 Uploading Phase (Algorithm 1). The DO encrypts the data file with symmetric encryption and secures the keys (data points) using an asymmetric algorithm for COs before uploading to CP.

Symmetric Key Construction for Encryption. PBCS utilizes coefficients from a Lagrange polynomial evaluated at zero, $L(0)$, to generate symmetric keys. These coefficients, derived from randomized sources and modulated by a decay parameter λ_d, maintain specific bits of entropy until they reach a predefined decay threshold. The process is designed to yield consistent keys, ensuring reproducibility with identical inputs.

Lemma 1. *Given data object d, Algorithm 2 generates a symmetric key k and θ shares (according to policy(d)) for k reconstruction with $O(N^2)$ complexity. The success probability with fewer than θ is $\approx \frac{1}{|\mathbb{F}_q|}$.*

Proof. Using the Lagrange polynomial defined in the Terminology Section 3.3, we evaluate the polynomial at zero as $L(0)$, which serves as the entropy seed for symmetric key generation. This value, $L(0)$, is then used as the input seed for a cryptographic hash function H: $K = H(L(0))$. The

ALGORITHM 2: KSS.ConstructKey

Input: Threshold θ
Output: Key k, datapoints *data_points*

1 Generate $\theta - 1$ random points $data_points \leftarrow (x_i, y_i) \in \mathbb{F}_q$
2 $L_x \leftarrow$ ConstructLagrangePolynomial(*data_points*)
3 seed $\leftarrow$ EvaluateAtZero(L_x)
4 key $\leftarrow$ H(seed)
5 **return** key k, *data_points*

key K, derived from $L(0)$, inherits its high entropy from the randomness of both x_i and y_i and the cryptographic hash function. Our scheme's security relies on symmetric encryption and a hash function. The computational complexity of constructing and evaluating the Lagrange polynomial-based symmetric generation is $O(N^2)$, where N refers to the number of shares and the probability of reconstructing the key without the required θ shares is negligible ($\approx \frac{1}{q}$, where q is the size of $|\mathbb{F}_q|$). □

Group Categorization Policy and Flexibility. PBCS strengthens group categorization and cryptographic security via dynamic threshold adaptation and periodic refreshes. Share compromise risks are assessed using a Bernoulli distribution with probability p, aggregated as $X \sim \text{Binomial}(n, p)$, where n is the number of shares [58]. The breach probability is: $P(\text{Breach}) = 1 - \sum_{k=0}^{\theta} \binom{n}{k} p^k (1-p)^{n-k}$, with θ as the threshold. Threshold adaptation adjusts n based on group size or risk, while periodic refreshes limit exposure of stale or compromised shares, ensuring security. The full specifications of the threshold adaptation and periodic refresh algorithms are provided in Appendix D.

Lemma 2 (Dynamic Threshold Adaptation). *The data owner can run Algorithm 10 to generate shares with a new threshold θ_{new} for the reconstruction of the same key k used in the encryption process.*

Proof. Consider the initial sharing polynomial $f^0(x) = s + \sum_{i=1}^{\theta_{init}-1} a_i x^i$ over $\mathbb{F}_q$, where s is the secret and θ_{init} the initial threshold. To adapt to a new threshold θ_{new}, the data owner samples $\theta_{new} - 1$ fresh random points $(x_i, y_i) \in \mathbb{F}_q$ and reconstructs the corresponding polynomial $L(x)$ using standard Lagrange interpolation (see Terminology 3.3). The updated seed is obtained by evaluating at zero, i.e., $L(0)$. The refreshed share set is then: $Updated_data_points = \{(x_i, y_i)\}_{i=1}^{\theta_{new}-1} \cup \{(0, L(0))\}$. Thus, the secret s (and hence the symmetric key k) is preserved, while reconstruction now requires at least θ_{new} valid shares. □

Lemma 3 (Periodic Refresh). *Every refresh time t timesteps, Algorithm 11 returns n new shares of threshold θ for the reconstruction of symmetric key k until t_{max}. After t_{max}, the key k will be forgotten according to decay factor λ.*

Proof. At predefined intervals t set by the CP, a renewal polynomial $R^t(x) = \sum_{i=1}^{t-1} R_i^t x^i$ is generated, with $R^t(0) = 0$ so that the original secret remains unchanged. Instead of using old shares, $n - 1$ new random data points $(x_i, y_i) \in \mathbb{F}_q$ are generated. Share renewal occurs as follows: $x_i^{new} = R^t(i)$. If the refresh time t has passed and $t < t_{max}$, new random data points are generated. Periodic refresh stops after the expiration time t_{max}. The data points then decay according to decay factor λ (see Definition 2), ensuring that the key k will be forgotten.

□

ALGORITHM 3: Preprocessing

Input: $SymEnc(d)$, AsymEnc(Shares(d)), $policy(d)$
Output: $Votes_{acc}, Votes_{del}, Val_{initial}$

Groups, H(Groups), $\theta_{acc}, \theta_{del} \leftarrow policy(d)$
foreach $group \in Groups$ **do**
  Generate a $nonce$[128] for freshness
  $GroupSignature \leftarrow$ Sign(H(GroupID‖nonce[128]), PrivateKey)
  Append $(GroupID, GroupSignature)$ to d_{acc}
end
$d_{del} \leftarrow \theta_{del} - 1$ random points $data_points$ in sets d_{del}
$L_{acc} \leftarrow ConstructLagrangePolynomial(d_{acc})$
$L_{del} \leftarrow ConstructLagrangePolynomial(d_{del})$
$Votes_{acc} \leftarrow$ Shares(d_{acc}) according to $policy(d)$
$Votes_{del} \leftarrow$ Shares(d_{del}) according to $policy(d)$
$Val_{initial} \leftarrow L_{del} \cap L_{enc/dec}$
return $Votes_{acc}, Votes_{del}, Val_{initial}$

The CP is the key for implementing these two attributes of PBCS. The data points representing the secret shares remain on the CP and are not distributed to the COs before the download phase is due to begin. This ensures that old shares cannot be used to reconstruct the key k after threshold adaptation and periodic refresh. Also, dynamic access revocation is enabled through these attributes.

Group-specific Public-Key Encryption for Polynomial Coefficients. PBCS supports using group-specific asymmetric encryption (PKI) to protect polynomial coefficients, ensuring that only authorized group members can decrypt them. This ensures data confidentiality: only users with the corresponding private keys can access the coefficients, thereby preserving data security (See Algorithm 12 in Appendix D).

Lemma 4. *Algorithm 12 encrypts polynomial coefficients using asymmetric encryption with $O(N)$ complexity, making the probability of unauthorized decryption negligible.*

Proof. Let $L(x) = \sum_{i=0}^{n} c_i x^i$ represent the Lagrange polynomial used for generating the symmetric encryption key k, with coefficients $\{c_0, c_1, \ldots, c_n\}$, where $c_i \in \mathbb{F}_q$. These coefficients are encrypted using asymmetric key encryption. The encryption for a specific group is given by: $c_i^{enc} = \mathrm{Enc}_{\mathrm{pub}}(c_i)$, where pub is the group's public key. The corresponding decryption uses the group-specific private key: $c_i = \mathrm{Dec}_{\mathrm{priv}}(c_i^{enc})$. The encryption and decryption operations have a computational complexity of $O(N)$, where N is the number of groups. The probability of decrypting the coefficients without the correct private key is negligible, assuming the security of the asymmetric encryption algorithm holds. □

4.2.2 Preprocessing Phase (Algorithm 3). The preprocessing phase occurs directly after uploading, when the data is outsourced to the provider. During this phase, both access control ($Votes_{acc}$) and deletion-based polynomials ($Votes_{del}$) are established. This step prepares the COs for downloading and deletion procedures by generating their respective polynomials. Additionally, the initial deletion validation ($Val_{initial}$) (run by the DO) is created to store the intersection points between the deletion polynomial and the original polynomial (from the upload phase). This mechanism will be used later to provide private proof of ephemerality to the COs. We reuse the same polynomial construction and threshold reconstruction primitives defined in the upload phase and Terminology 3.3; preprocessing only instantiates them for $Votes_{acc}$, $Votes_{del}$, and $Val_{initial}$.

ALGORITHM 4: Access-Controlled Download

Input: $Votes_{\text{acc}}$
Output: Access decision (0 or 1)

$L_{access} \leftarrow ConstructLagrangePolynomial(V_{acc})$
return $L_{access} \equiv L_{acc}$

ALGORITHM 5: Data Recovery

Input: $SymEnc(d,k)$, $AsymEnc(k_{pub_g}, data_points)$, Participation matrix m, threshold θ
Output: d

$data_points \leftarrow AsymDec(AsymEnc(k_{pub_g}, data_points), k_{pri_g})$
$k \leftarrow Key_Reconstruction(m, data_points, \theta)$ [1]See Algorithm 15 in Appendix D.
$d \leftarrow SymcDec(SymEnc(d,k),k)$
return d

4.2.3 Download Phase (Algorithms 4 and 5). The access control polynomial $L_{\text{access}}(x)$ is designed to validate access requests denoted as $Votes_{\text{acc}}$. It ensures that only authorized groups or co-owners can access the data based on predefined policies $policy(d)$. The polynomial uses tuples of group identifiers and cryptographic signatures to verify participation, granting access only when policy-defined thresholds are met.

Access Control Verification: Each access request (vote) is authenticated using cryptographic hashes and signatures. $L_{\text{access}}(x)$ verifies if the votes $Votes_{\text{acc}}$ comply with $policy(d)$ by matching the expected values. This ensures that access decisions are based on confirmed, legitimate group consensus.

Lemma 5. *Given $Votes_{acc}$ created according to $policy(d)$ of data d, with complexity $O(N^2)$, Algorithm 4 outputs* 1 *if the group is authorized to access d and* 0 *otherwise.*

Proof. Let $G = \{(ID_i, Sig_i)\}_{i=1}^{n}$ represent the set of tuples, where $ID_i \in \mathbb{Z}$ denotes the unique identifier for the i^{th} group and $Sig_i \in \mathbb{Z}_N$ is the cryptographic signature, signed with its asymmetric private key. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be a cryptographic hash function, where $\mathbb{Z}_q$ is a finite field of prime order q. Construct the access control polynomial $L_{access}(x)$ by interpolating the points (x_i, y_i) using standard Lagrange interpolation (see Terminology 3.3), where $y_i = H(ID_i \,\|\, Sig_i \oplus nonce[128])$. To validate access, a group provides its ID and Sig, deriving a verification point $(x_i, \hat{y}_i)$, with $\hat{y}_i = H(ID \| Sig \oplus nonce[128])$, where *nonce* serves as a seed to provide randomness. The system then evaluates $L_{access}(x_i)$ and checks if: $L_{access}(x_i) \equiv \hat{y}_i \mod q$. If equality holds, access is granted; otherwise, it is denied. A unique Lagrange polynomial passes at least through θ points (x_i, y_i). If the θ is not reached, then $L_{access}(x) \neq L_{acc}(x)$. Constructing and evaluating $L_{access}(x)$ is $O(N^2)$, where N refers to the number of groups. The probability of forging valid $(x_i, \hat{y}_i)$ without the correct combination of $(ID, Sig \oplus nonce)$ is negligible (See Algorithm 17 for more details). □

Data Recovery: Once access is granted, authorized **co-owners** (**COs**) proceed with the recovery phase. During this phase, the asymmetric private keys of authorized co-owners are used to decrypt the coefficients, ensuring compliance with the verified policies. After successful coefficient decryption, the symmetric key reconstruction type is determined (singular or collaborative) based on the participation matrix m, which defines group thresholds and reconstruction requirements. Data is then recovered as in Algorithm 5.

Decryption Process Leveraging Public-Key Encryption and Group-Specific Access. Upon successful decryption using the group-specific asymmetric private key, the original polynomial

$$\begin{array}{c} \\ User_1 \\ User_2 \\ \\ User_n \end{array} \begin{array}{c} \begin{array}{cccc} Group_1 & Group_2 & \dots & Group_n \end{array} \\ \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{bmatrix} \end{array} \begin{array}{c} V \\ \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \end{array}$$

Fig. 2. Group membership matrix m.

$L_{enc}(x)$ used for encrypting the data is accurately reconstructed, enabling the secure decryption of encrypted coefficients stored within data objects. This ensures that only authorized group members possessing the correct private keys can access and decrypt the data for further use without CP intervention (see Algorithm 13 in Appendix).

Lemma 6. *Algorithm 13 decrypts polynomial L_{enc}'s coefficients using the group's private key with $O(N)$ complexity.*

Singular/ Collaborative Lagrange Polynomials Decryption-based Matrix Approach. We address the generation of Lagrange polynomials for decryption in a matrix-based approach in **singular** and **collaborative** scenarios. If a user has multiple IDs or group memberships, the polynomial generation becomes collaborative, ensuring the key cannot be reconstructed with fewer than the threshold θ as defined by $policy(d)$. This design guarantees that users with multiple group memberships have ***only one share***. The matrix m validates memberships and determines the key reconstruction type. Access is granted if user memberships and group intersections satisfy verification conditions: $V = \{v_1, v_2, ..., v_n\}$, where $v_i \in \{0, 1\}$, as illustrated in Figure 2.

Lemma 7 (Decryption Polynomial Generation). *Given participation matrix m, data points shares, and threshold θ, Algorithm 15 returns the data object from shares, if there at least θ unique shares with $O(N^2)$ complexity. The success probability with fewer than θ shares is $\approx \frac{1}{|\mathbb{F}_q|}$.*

Proof. The polynomial generation algorithm determines whether decryption follows a *singular* or *collaborative* construction based on group participation and threshold requirements. For a group G, the singular polynomial $L_{\text{singular}}(x)$ is obtained by interpolating the points $\{(x_i, y_i)\}_{i \in G}$ using standard Lagrange interpolation (i.e., Section 3.3), and reconstructs the secret when the group-specific threshold θ_i is satisfied. Thus, each group can independently reconstruct the key once its own threshold is met.

To ensure collective access, multiple groups must conjunctively satisfy their respective thresholds. For users $\{U\}$ belonging to different, possibly overlapping groups, a collaborative polynomial $L_{\text{collab}}(x)$ is constructed by interpolating the unique shares contributed by these users, again using the standard Lagrange mechanism. This guarantees that each user contributes exactly one share and that reconstruction is possible only when the global threshold θ defined by $policy(d)$ is met.

Security follows from the hardness of polynomial interpolation with fewer than θ valid shares. With insufficient points, the reconstructed polynomial deviates from the correct one, with error bounded by the Lagrange remainder [14]:

$$E(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \quad \xi \in [x_0, x_n].$$

Hence, reconstructing the secret without at least θ shares succeeds with probability at most $\approx \frac{1}{|\mathbb{F}_q|}$.

The construction and evaluation of the corresponding interpolation polynomial have complexity $O(N^2)$, where N denotes the number of participating shares or groups. □

ALGORITHM 6: Deletion

Input: $Votes_{\text{del}}$
Output: Deletion decision (0 or 1)

1 $L_{delete} \leftarrow ConstructLagrangePolynomial(V_{del})$
2 **return** $L_{deletion} \equiv L_{del}$

Symmetric Key Reconstruction for Decryption. The PBCS employs Lagrange polynomial coefficients, initially used for encryption, to reconstruct symmetric keys for decryption after asymmetric private keys are used to decrypt the data, ensuring secure access to encrypted data by authorized COs.

Lemma 8. *During the validity period, Algorithm 5 returns data object d given the encrypted data object d′, shares, participation matrix m, and threshold θ, and has $O(N^2)$ complexity.*

Proof. First, we decrypt the *data_points* using the group-specific private key. We recover the symmetric key k used to encrypt the data object d using the data points. Finally, we can locally decrypt the data object with k. Since we are still in the validity period, the data points will be sound and not decayed. The complexity of constructing and evaluating the Lagrange polynomial-based symmetric key generation is $O(N^2)$, where N refers to the number of groups. □

We expand upon what happens after the validity (i.e., set by the policy) period below in the *Forgetting Phase* in Section 4.2.5.

4.2.4 Deletion Phase (Algorithm 6). In support with GDPR *(Recitals* 65, 66, *and Article* 17*)*, our scheme empowers COs to initiate data deletion through validated requests $Votes_{\text{del}}$, supporting data sovereignty and privacy. This deletion is controlled via a polynomial $D(x)$, which enables secure erasure when a threshold Δ is met. The deletion polynomial $L_{delete}(x)$ over a finite field $\mathbb{F}_q$ is defined as: $L_{delete}(x) = \sum_{k=1}^{T} d_k x^k + d_0 \mod q$, where $d_0, d_1, \ldots, d_T$ are coefficients encoding deletion criteria, with x as the variable for progression and T as the deletion threshold. Data erasure is triggered when: $L_{delete}(x^*) = \Delta \mod q$, where x^* denotes the condition met, and Δ represents the activation for deletion.

Lemma 9 (Data Object Deletion). *Given $Votes_{del}$ created according to policy(d) of data d, Algorithm 6 outputs* 1 *if the data object d is deleted and* 0 *otherwise, with the complexity of $O(N)$.*

Proof. Once the deletion polynomial D is constructed from the COs' deletion request, if $D(0)$ matches the stored deletion polynomial $L_{del}(0)$, the CP will proceed to delete the data object d and return 1. The complexity of this deletion process is $O(N)$, where N represents the number of data objects in the cloud storage. □

Voting Techniques. We have used two criteria, such as majority voting and veto rights, to construct the deletion polynomial. These two approaches provide a flexible method for data deletion (see Figure 3).

Lemma 10 (Deletion Polynomial-based Majority Voting). *Algorithm 7 constructs a deletion polynomial $D(x)$ if $Votes_{del}$ meet the majority threshold with $O(N^2)$ complexity.*

Proof. Let n co-owners cast $Votes_{\text{del}} = \{v_1, v_2, ..., v_n\}$, where $v_i \in \{0, 1\}$ for each $i = 1, 2, ..., n$, indicating a vote against (0) or for (1) deletion. A deletion polynomial $D(x)$ is constructed when the sum of affirmative votes reaches or surpasses the majority threshold $T_m = \lceil \frac{n}{2} \rceil$. Define the

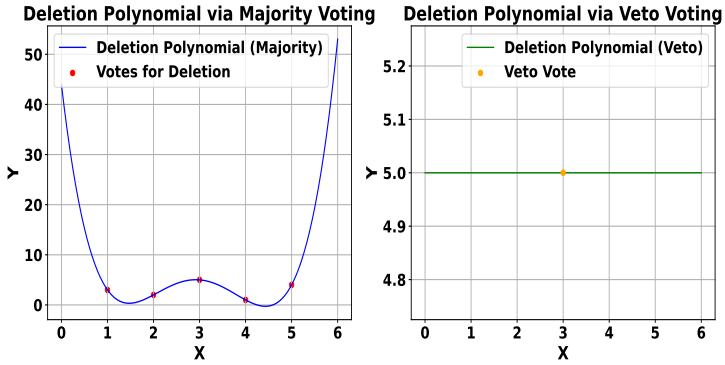


Fig. 3. Deletion polynomial-based voting mechanisms.

ALGORITHM 7: Majority-Based Deletion Polynomial

Input: $\text{Votes}_{\text{del}}$, Majority
Output: Deletion polynomial $D(x)$ or null

```
if |Votes_del| ≥ Majority then
    for i = 1 to Length(Votes_del) do
        if Votes_del[i] = 1 then
            D(x) ← D(x) + (GroupPoints[i] · LagrangeBasis(i, GroupPoints)) mod q
        end
    end
    return D(x)
end
return null
```

activation function for constructing the deletion polynomial $D(x)$ as follows:

$$D(V) = \begin{cases} \sum_{i=1}^{n} v_i \cdot (x_i, y_i), & \text{if } \sum_{i=1}^{n} v_i \geq T_m, \\ \text{Null}, & \text{Otherwise} \end{cases}$$

Here, (x_i, y_i) represent the points contributed by affirmative votes ($v_i = 1$) to construct $D(x)$, so the deletion polynomial is formed only once the policy's required majority is reached. If a co-owner is later revoked, the DO updates $policy(d)$ (membership m/eligible keys) to exclude that identity and triggers a refresh/re-key so that only the updated authorized set can cast valid votes or reconstruct k; any prior shares/votes from the revoked party are rejected under the new policy. In large groups, this refresh incurs a one-time overhead proportional to the affected shares/groups.

□

4.2.5 *Forgetting Phase.*

Lemma 11 (Veto Voting Right). *Algorithm 8 constructs a deletion polynomial $D(x)$ if the hashed secret $H(s_v)$ matches the input and null otherwise with $O(1)$ complexity.*

Proof. Assume a designated co-owner (i.e., assigned by DO) possesses a veto power encapsulated by a hashed secret s_v, corresponding to the polynomial's zero point $f(0)$, enabling this co-owner

ALGORITHM 8: Veto Deletion Polynomial

Input: Hashed_Secret, vetoPoint
Output: $D(x)$ or `null`

```
if ValidateHash(Hashed_Secret) then
    return (vetoPoint · LagrangeBasis(0, {vetoPoint})) mod q
end
return null
```

to satisfy the deletion condition directly through a singular action. Let $Votes_{\text{del}} \equiv H(s_v)$, where $H(s_v)$ denotes the hashed secret acting as the veto power, where $s_v = f(0)$ is the secret at the polynomial's zero point. The deletion polynomial $D(x)$ is influenced by introducing a term reflecting the veto power: $D_{veto}(x) = D(x) + H(s_v) \cdot \delta(x)$, where $\delta(x) = 1$ for $x = 0$ and 0 otherwise, effectively acting as a Kronecker delta function [46] to simulate the inclusion of $H(s_v)$ in $D(x)$ at $x = 0$. This modification ensures that activating veto power via $H(s_v)$ immediately satisfies the deletion condition, demonstrating the veto right's unique ability to trigger the deletion mechanism independently of the majority consensus. □

The security reduces the difficulty of reaching the majority threshold $Votes_{\text{del}}$ and breaking the hashed secret $H(s_v)$. If the majority threshold is not met, the polynomial cannot be constructed, preventing deletion from proceeding. Similarly, if the hash function is secure, unauthorized triggering of the veto is infeasible. The complexity of constructing the deletion polynomial is $O(N^2)$, and the complexity of checking the veto condition is $O(1)$. The probability of unauthorized deletion without the majority or veto is negligible.

Following key decay, COs may exercise their RTBF rights by requesting deletion validation from the CP, which confirms the secure erasure of the data. As detailed in Section 4.1.1, the key's validity diminishes over time under both entropy-driven passive decay and policy-enforced active decay. When validity drops below the threshold θ, decryption becomes unfeasible, i.e., $E(x, f_{\text{passive}}(t), f_{\text{active}}(t)) < \theta \Rightarrow \{\text{Forgetting}\}$. In the conjunctive decay model, this occurs when the required individual thresholds θ_i or the collective threshold θ_{collab} are no longer satisfied. As time progresses, if $E(x)$ with $f_{\text{passive}}(t)$ and $f_{\text{active}}(t)$ evaluates below θ, it signals sufficient decay for the forgetting phase: $\lim_{t\to\infty} E(x, f_{\text{passive}}(t), f_{\text{active}}(t)) = 0$. **If passive decay progresses prematurely, temporary reconstruction is possible if $\theta + m$ shares ($m > 0$) remain intact due to differing decay rates. However, as all shares eventually decay, key irretrievability is enforced, ensuring secure deletion.**

Intersection Between Original and Deletion Polynomials. The proposed scheme employs polynomial algebra to ensure data deletion by analyzing intersections between the original encryption polynomial, $E(x)$, and the deletion polynomial, $D(x)$. Secure deletion is verified once the differential polynomial $E'(x)$, derived from the original encryption polynomial $E(x)$ and the deletion polynomial $D(x)$, demonstrates roots within the operational domain (i.e., presence of solutions of intersection). This indicates new intersection sets distinct from previous overlaps between $E(x)$ and $D(x)$ (i.e., $\neq \text{Val}_{\text{initial}}$), thereby confirming the successful completion of the deletion process (See Figure 4). This private validation is performed by the DO and authorized COs (who possess the required shares/keys) and provides evidence of cryptographic erasure, i.e., that key reconstruction becomes impossible under the θ-threshold after an authorized deletion decision. It does not certify physical wiping of all ciphertext copies by the CP; rather, it certifies that any retained ciphertext is unusable because the corresponding key material is irrecoverable (i.e., cryptographic forgetting only), supporting RTBF objectives.

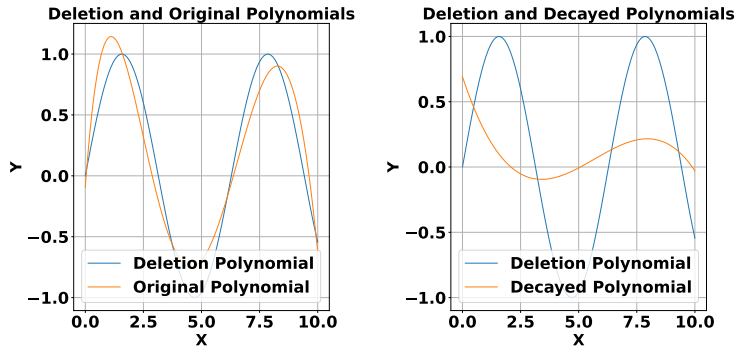


Fig. 4. Deletion validation: intersection between original and deletion polynomials within different periods.

ALGORITHM 9: Deletion Validation

Input: $Shares(d), Votes_{del}$
Output: 0 or 1 indicating deletion validation

Let $E(x)$ be the $Shares(d)$ and $D(x)$ be $Votes_{del}$
$E'(x) \leftarrow E(x) \cdot f(t)$ ▷ *[r]Apply decay to $E(x)$ with $f(t)$
$m \leftarrow \deg(E)$, $n \leftarrow \deg(D)$, $R(x) \leftarrow 0$
for $k = 0$ **to** $\max(m, n)$ **do**
  $r_k \leftarrow E_k - D_k$ ▷ *[r]Subtract coefficients of $D(x)$ from $E(x)$ $R(x) \leftarrow R(x) + r_k x^k$
end
roots $\leftarrow$ FindRoots(R, domain) ▷ *[r]Find roots of $R(x)$ in the specified domain **if** *roots are not empty* **then**
  return 1 ▷ *[r]Intersection confirmed, deletion criteria met
else
  return 0 ▷ *[r]No intersection, deletion criteria not met
end

Lemma 12 (Deletion Validation). *If data object d has been deleted, Algorithm 9 returns* 1 *if the data is no longer accessible as a key is no longer recoverable with a complexity of* $O(N \log N)$ *and a probability of failure* $\approx \frac{1}{|\mathbb{F}_q|}$.

Proof. Define $E(x)$ as the polynomial for encryption/decryption and $D(x)$ as the polynomial for secure data deletion, with degrees $deg(E) = m$ and $deg(D) = n$. Construct the differential polynomial $R(x) = E(x) - D(x)$, where $deg(R) = \max\{m, n\}$, assuming $E(x) \neq D(x)$ with distinct leading coefficients. We analyze $R(x)$ to verify intersections within a specified domain to match the intersected points between initial proof generated (Algorithm 3) earlier and the latest one upon request, ensuring at least one root of $R(x)$ exists to validate the deletion process: $R(x) = \sum_{k=0}^{\max\{m,n\}} r_k x^k$, where r_k are the coefficients from subtracting $D(x)$ from $E(x)$. The set $S = \{x \in \mathbb{C} \mid R(x) = 0\}$ represents the intersection points. The existence of S confirms that deletion criteria are met, based on consensus for roots $\{x_1, x_2, \ldots, x_k\}$ at time t, suggesting $E(x_i) = D(x_i)$. The transformation of $E(x)$ due to the key decay function $f(t)$ (with domain $t \in \mathbb{R}^+$), defined as $e^{-\lambda t}$ where λ is the decay constant, exponentially reduces the key's effectiveness: $E'(x) = E(x) \cdot$

$f(t)$ and consequently, $R'(x) = E'(x) - D(x)$. As t approaches infinity, $\lim_{t\to\infty} E'(x) = 0$. This ensures the data transitions to a 'forgotten' state. The intersection of $E(x)$ and $D(x)$ within the domain confirms irreversible, secure deletion, initiating the forgetting phase under controlled conditions. Our scheme's security reduces the difficulty of finding the roots of $R(x)$. The complexity of computing polynomial intersections is $O(N \log N)$, where N represents the number of data points. The probability of falsely confirming deletion without an actual intersection is negligible. □

4.3 Formal Analysis

4.3.1 Ideal Functionality $\mathcal{F}_{PBCS}$. In the ideal world, **Trusted Third Party** (**TTP**) securely manages the lifecycle of d through the phases:
1) Upload: TTP receives d with policies and an expiration time T_{exp} from DO. **2) Deletion:** TTP receives a request from the COs, if the policy satisfied then $T_{\text{exp}} = now$. **3) Download:** TTP outputs d if the COs requested it after fulfilling the provided policy P before T_{exp}.

Theorem 4.1 (Correctness). *PBCS is correct (Definition 5).*

Proof. For any uploaded object d, the downloaded object satisfies: $\text{Download}(\text{Upload}(d), m, \theta) = d$, with negligible failure probability $P[\text{failure}] \leq \epsilon$, where ϵ is the negligible failure probability, based on $\mathbb{F}_q$ under a semi-honest CP. Correctness is maintained even in the presence of malicious COs, as they cannot reconstruct k of data d without meeting the policy-defined threshold. Polynomials $L_{\text{access}}(x)$ and $L_{\text{delete}}(x)$ validate voting shares, ensuring that only authorized actions—such as data access or deletion—are permitted. The cryptographic integrity of these polynomials, grounded in the entropy-preserving properties of SSSS [68], prevents malicious COs from forging or providing random votes. Furthermore, the collaborative decay mechanism ensures that k becomes irretrievable as $K(t) < \theta_{\text{collab}}$ (see Section 4.1.1). Furthermore, only authorized COs can validate and execute deletion requests, preserving correctness under adversarial conditions, as proven by lemmas 1 to 12. □

Theorem 4.2 (Privacy of Inputs). *PBCS is Λ-secure (Definition 6).*

Proof. **(1) Real World Scenario:** In practice, the DO, CP, and COs operate without a TTP, employing distributed cryptographic protocols that reflect $\mathcal{F}_{PBCS}$'s secure management. In the PBCS, DOs encrypt data objects d using *SymEnc* with *AsymEnc*-managed keys, rendering them indistinguishable from the CP, whether d or its altered version d'. COs access and deletion based on enforced policies, thus preserving indistinguishability across all operations.
(2) Simulation for CP: Assume real-world operation with data object d, actual *data_points*, and initial validation $Deletion_{val}$. Replace d and *data_points* with a random data object d' with random points *random_data_points*, and random validation $Deletion_{random_{val}}$ in the simulation. The preprocessing, access control, deletion, and download phases follow the protocol. The $Deletion_{val}$ has insufficient points for the CP to gain information about the symmetric key. Thus, the $Deletion_{val}$ is indistinguishable from $Deletion_{random_{val}}$ from the CP side, as they both look like random points. Since the data is encrypted using symmetric encryption with keys secured by the public-key encryption algorithm and considering the CP does not possess the decryption keys, $SymEnc(d)$, $AsymEnc(data_points)$, and $SymEnc(d')$, $AsymEnc(random_points)$ are computationally indistinguishable under the IND-CPA security [12] from the CP (See Theorems 1, 4, 6, 8, and 12). The CP cannot distinguish the simulation from the real-world run ($\Pr[V_{\text{real}} \neq V_{\text{sim}}] \leq \epsilon$, where ϵ is negligible.
(3) Simulation for COs: A malicious COs ($\mathcal{A}_{\text{CO}}$) may attempt to compromise the system by bypassing critical security mechanisms. Specifically, they could aim to: **(i)** gain unauthorized access to d by circumventing the access control polynomial $L_{\text{access}}(x)$ using invalid or insufficient ($< \theta$)

shares; or **(ii)** execute unauthorized deletion of d without adhering to the voting thresholds enforced by $L_{\text{delete}}(x)$. These actions reflect malicious intent to subvert policy-defined security measures and exploit system vulnerabilities. In the simulation, actual shares $Shares_{\text{acc}}, Shares_{\text{del}}$ are replaced with random shares $Shares_{\text{random}_{\text{acc}}}, Shares_{\text{random}_{\text{del}}}$, which appear uniformly random in $\mathbb{F}_q$, making them indistinguishable from the real-world shares. Access or deletion is only granted when the shares satisfy the policy-defined polynomials $L_{\text{access}}(x)$ or $L_{\text{delete}}(x)$. Without meeting these threshold requirements, the cryptographic entropy of polynomial-based secret sharing ensures that the actual data object d remains inaccessible. Even when a co-owner belongs to multiple groups, the participation matrix m enforces additional constraints, requiring valid shares for collaborative reconstruction (see Section 4.2.3). The probability of success for malicious co-owners is negligible, as SSSS guarantees [68] that fewer than θ valid shares reveal no information about the secret: $\mathbb{P}[\mathcal{A}_{\text{CO}} \text{ succeeds}] \leq \frac{1}{|\mathbb{F}_q|}$. The simulation remains indistinguishable from real-world execution, ensuring no unauthorized access or deletion occurs (See Theorems 5, 7, and 9).

□

4.4 Adversarial Models Analysis

PBCS mitigates a malicious CP that skips all computations via passive decay and PKI-encrypted shares to ensure key irretrievability.

Setup: Key shares are encrypted under the co-owners' PKI. Shares are not static values but links that resolve to dynamic, entropy-driven outputs.
Challenge: A malicious CP can skip all computations, including access control, deletion, deletion validation, and active decay, while attempting to: (*i*) reconstruct the key; (*ii*) retain encrypted data copies.
Winning Condition: Decrypting key shares without PKI is infeasible, and reconstruction remains improbable: $\mathbb{P}[\text{CP reconstructs } K] \leq \frac{1}{|\mathbb{F}_q|}$. Even without active decay, passive decay continues: (x_i', y_i'). Since keys follow entropy-driven, non-static links, snapshots become ineffective, leading to irreversible forgetting despite a malicious CP.

PBCS prevents eavesdropping and inference attacks using uniform padding techniques through operation timing obfuscation.

Setup: Let $P(T_d)$ be the padded transmission time (i.e., adding fixed time delays) for data object d. Padding ensures that access and deletion times are indistinguishable: $P(T_d) = P(T_{acc}) = P(T_{del})$.
Challenge: The adversary $\mathcal{A}$ attempts to infer operation types (access or deletion) or deletion schedules by analyzing timing/statistical information (e.g., response times or observable request patterns).
Winning Condition: $\mathcal{A}$'s probability of correctly guessing the operation type from timing is bounded: $\mathbb{P}[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \epsilon$, where ϵ is negligible. Any residual leakage from higher-level traffic patterns (e.g., refresh/validation cadence) is an operational side channel that is mitigated by batching, adding bounded random scheduling variation (i.e., jitter), and avoiding per-object timing signals.

PBCS mitigates collusion attacks between CP and COs through PKI safeguards for decryption and KSS for secure key reconstruction.

Setup: Co-owners hold private keys under PKI to decrypt encrypted data points and possess shares required for symmetric key reconstruction, governed by the participation matrix m and policy-defined collaborative threshold θ.

Challenge: An adversary $\mathcal{A}$, consisting of the CP and a subset of malicious co-owners, attempts to: (*i*) bypass access enforcement to retrieve d without fulfilling policy conditions; or (*ii*) evade deletion compliance by retaining snapshots of d, nullifying decay mechanisms.
Winning Condition: Collusion attacks may bypass access and deletion controls, but unauthorized decryption and key reconstruction remain secure. PKI enforces decryption limits through computational hardness assumptions [12], while collaborative threshold enforcement ensures key recovery requires θ shares, with success probability for fewer shares bounded by $\mathbb{P}[\mathcal{A} \text{ reconstructs } K] \leq \frac{1}{|\mathbb{F}_q|}$ (Theorem 7).

5 Results and Insights

A prototype was developed to evaluate its efficacy in secure, conjunctive audience-specific data management and GDPR alignment. The prototype utilized AES-256-GCM for encryption and RSA-OAEP with a 2048-bit key and SHA-256, following NIST standards [18, 23, 25, 37, 53]. Testing was conducted on a MacBook with a 2.3 GHz Intel Core *i*5 and 8 GB RAM, simulating a multi-tier cloud storage environment to examine encryption, access control, secure deletion, and conjunctive key decay. Note that the study utilized scientific draft document PDFs as the data source. This section details (*i*) Integration with KMS, (*ii*) Evaluation Metrics, (*iii*) Experimental Results, and (*iv*) Limitations.

5.1 Integration with Key Management Services (KMS)

PBCS is compatible with standard envelope-encryption deployments. In our prototype, each data object is encrypted locally with an AES-256-GCM data-encryption key k (DEK), while co-owner shares/points and voting evidence are protected using RSA-OAEP under the co-owners' PKI. In production, this PKI layer can be instantiated using cloud KMS-managed asymmetric keys (e.g., AWS KMS [4]) without changing PBCS: share/point blobs can be encrypted to KMS-backed public keys, and decryption becomes IAM-gated and auditable; vote messages can likewise be signed with KMS-backed keys to provide integrity and audit trails. Entropy sourcing and decay remain within PBCS; the KMS provides key custody/auditing (and optional wrapping/signing) but does not supply entropy or affect the policy-defined validity window and irrecoverability of key k.

5.2 Evaluation Metrics

Reliability of Results. Our system exhibits probabilistic data lifetimes, yet extensive testing across varied configurations consistently confirmed reliable data forgetting in the conjunctive scheme.
Key Decay Guidelines (Table 2). Parameters were empirically selected to achieve data inaccessibility over periods ranging from days to weeks, illustrating various active-decay functions (Linear, Exponential, and Sigmoidal) in a collaborative environment. Decay intervals range from long-term to short-term access, using metrics tailored for cloud collaboration. Parameters are categorized by decay type, with key validity spanning weeks to hours, and adjusted for sensitivity (α_i), importance (w_i), and stability (μ_i) to ensure secure and predictable decay behavior.
Error Corrections. We applied error correction ($e_{0\to1}$) using the remainder theorem [22] in polynomial division to improve cryptographic reliability. If $P(x)$ is the interpolated polynomial and $G(x)$ is the generator polynomial, errors are detected by a non-zero residue $r(x) = P(x) \mod G(x)$. By evaluating the $P(x)$ derivative, errors are located and corrected. Testing showed an error rate of $e_{0\to1} = [0\%, 0.002\%]$, ensuring high reliability [85].

Table 2. Parameters Space for Conjunctive Key Decay Used in Study

Parameter	Low Decay	Moderate Decay	High Decay	Exponential Decay	Sigmoidal Decay
$\lambda(\mathbf{hr}^{-1})$	$[0.001, 0.002]$	$[0.006, 0.020]$	$[0.040, 0.080]$	$[0.01, 0.1]$	$[0.001, 0.1]$
T	$[3, 4]$ weeks	$[2, 7]$ days	$[12, 24]$ hours	$[1, 7]$ days	$[1, 5]$ days
$\Delta\lambda(\mathbf{hr}^{-1})$	$+0.01\%$	$+0.33\%$	$+20\%$	$+0.21\%$	$+1.67\%$
$K(t)$ **(Figure 5)**	Long-term access	Medium-term access	Short-term access	$K_0 \cdot e^{-\lambda t}$	$\frac{K_0}{1+e^{-\lambda(t-T_{\text{mid}})/s}}$
$\mu_{50}(\mathbf{min}^{-1})$	≤ 20	≤ 10	≤ 5	Sensitive to initial value	Slow start, rapid midpoint increase
$\mu_{100}(\mathbf{min}^{-1})$	$[80, 100]$	$[45, 60]$	$[10, 20]$	Decay stabilizes after initial drop	Stable after midpoint $s \in [0.1, 10]$
$\mu_{200}(\mathbf{min}^{-1})$	$[140, 160]$	$[100, 115]$	$[30, 50]$	High stability across time	Stable after midpoint
θ	$\theta = 0.10$ (Longer access)	$\theta = 0.05$ (Moderate access)	$\theta = 0.01$ (Shorter access)	$\theta \in [0.02, 0.05]$ (Rapid unrecoverability)	$\theta \in [0.03, 0.05]$ (Gradual loss, steep decay)
α_i	$\alpha = 1.0$ (Standard sensitivity)	$\alpha = 1.5$ (Moderate sensitivity)	$\alpha = 2.0$ (High sensitivity)	Adjusts with policy	Adjusts with policy
w_i	$w = 1.0$ (Standard importance)	$w = 1.5$ (Moderate importance)	$w = 2.0$ (High importance)	Adjusts with policy	Adjusts with policy

Legend:(1) λ – Key decay. **(2) s** – Sigmoidal steepness. **(3)** K_0 – Initial key value. **(4) T** – Key validity. **(5)** $\Delta\lambda$ – Rate of decay increase. **(6) K(t)** – Key effectiveness. **(7)** μ – Stability across value sets. **(8)** θ – Threshold. **(9)** α_i – Decay sensitivity factor. **(10)** w_i – Weight representing share importance.

5.3 Experimental Results

Our experiments targeted two areas: (*i*) Decay Analysis and (*ii*) Performance Evaluation. Additional results are discussed in Appendix C.

5.3.1 Decay Analysis. This analysis examines how key ephemerality affects data access control within the conjunctive framework of PBCS, which is essential for maintaining security and compliance in collaborative cloud environments. By evaluating 100 groups and 1,000 co-owners, we explored distinct decay models tailored to address varied access and expiration scenarios.

(a) *Key Effectiveness Over Time (Figure 5):* This study evaluated runtime variations across key effectiveness levels from 70% to 20%, aligning with the conjunctive scheme's efficiency needs. Linear decay provided a steady, controlled decline suitable for gradual access reduction. Exponential decay enabled rapid inaccessibility, which is ideal for swift data obsolescence. Sigmoidal decay showed an *S*-shaped curve, with an initial period of resistance and a rapid decline, supporting adaptable security protocols. The runtime analysis (minimum, maximum, average, and deviations) shows consistent performance, with all decay models maintaining stability as effectiveness decreases.

(b) *Group Dynamics and Key Ephemerality (Figure 6(a)):* The study demonstrates how decay functions influence access within overlapping group memberships critical to the conjunctive scheme's operation. **Linear Decay** ($K_{\text{linear}}(t) = \max(0, 1 - \lambda t)$) ensures key effectiveness remains non-negative as it declines. This rapid turnover model reaches an empirical inaccessibility threshold of 20% within 2 days, making it ideal for time-sensitive tasks that require immediate access revocation. **Exponential Decay** ($K_{\text{exp}}(t) = e^{-\lambda t}$) provides a gradual phase-out, extending accessibility over time, which aligns with tasks involving progressive security tightening. **Sigmoidal Decay** ($K_{\text{sig}}(t) = \frac{1}{1+e^{-\lambda(t-T_{\text{mid}})/s}}$) is optimal for workflows requiring an initial stability phase, followed by

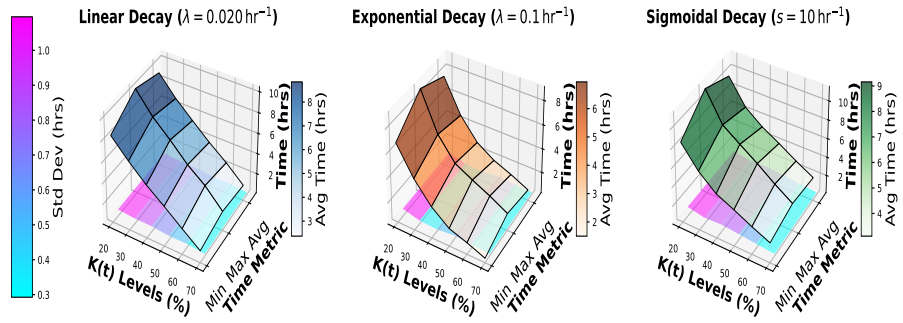
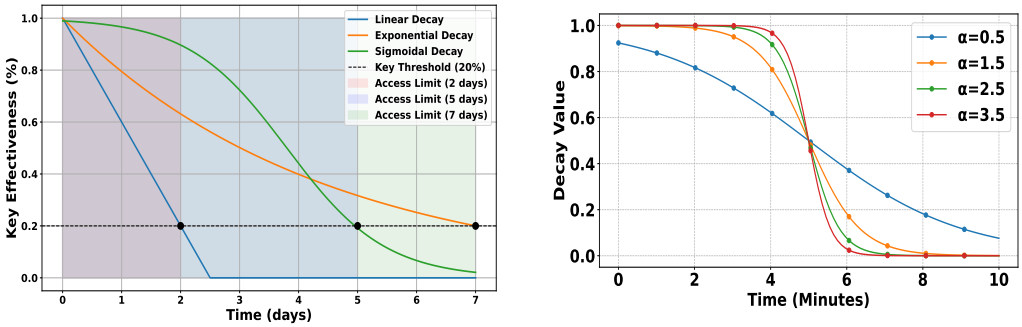


Fig. 5. Run-time statistics for various decay factors influencing key effectiveness ($K(t) \in [70\%, 20\%]$).



(a) Key effectiveness over time with decay rates, highlighting conjunctive access limits in 2 to 7 days.

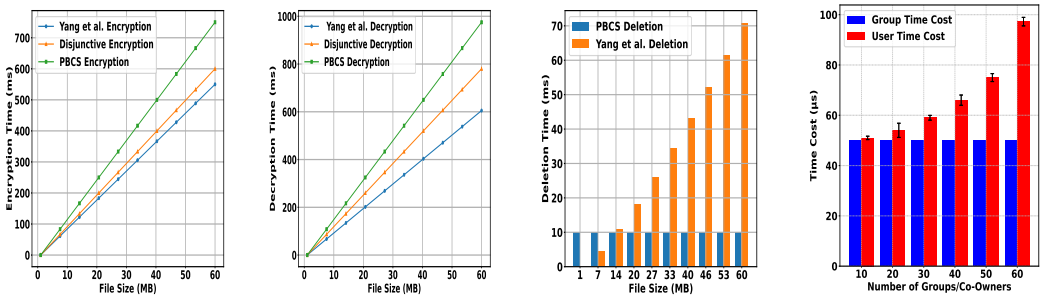
(b) Impact of sigmoidal decay on key sensitivity across different sensitivity levels (α_i) over time.

Fig. 6. Decay behavior and sensitivity analysis in the proposed forgetting scheme.

rapid access restriction as the key approaches expiration. This curve effectively balances accessibility and security, particularly for transitional data retention phases. This analysis affirms that decay rates are critical to the conjunctive scheme's ability to enforce dynamic and controlled access, with distinct impacts on groups with multi-level access.

(c) *Sensitivity Levels Impact on Decay Functions (Figure 6(b)):* Sensitivity levels, α (0.5, 1.0, 1.5, and 2.0), play a critical role in adjusting decay rates. In linear and exponential models, increasing α accelerates decay, promoting rapid data expiration. However, in the sigmoidal model, defined as $K(t) = \frac{1}{1+e^{-\alpha\lambda(t-c)}}$, a higher α initially slows decay during the *Early phase* ($t < 5$) but results in a sharper and faster decay after the midpoint at $t = 5$ (*Middle phase*). The rate of change is expressed as: $\frac{dK(t,\alpha)}{dt} = \frac{\alpha\lambda e^{-\alpha\lambda(t-c)}}{(1+e^{-\alpha\lambda(t-c)})^2}$. For $t < 5$, lower α values lead to faster decay, while for $t > 5$, higher α values cause a more rapid decline. This dynamic provides flexibility in managing group-specific expiration rates, which enhances adaptability.

5.3.2 Performance Evaluation. This section evaluates the conjunctive scheme's efficiency, focusing on computational time and scalability in complex environments.



(a) Encryption Comparison. (b) Decryption Comparison. (c) Deletion Comparison. (d) Validation Comparison.

Fig. 7. Performance evaluation of various time costs across different schemes and validation time in PBCS.

Computational Time (Figure 7). This assessment focused on the operational time through the phases, including encryption, decryption, deletion, and validation.

(a) *Encryption Time (Figure 7(a)):* The time for the three schemes increases as the file size grows. The PBCS scheme shows the highest encryption time across all file sizes, peaking at 750 ms for 60 MB, due to its $O(n^2)$ complexity from Lagrange interpolation for seed generation and dual encryption layers for data and policy enforcement. Yang et al. perform faster than PBCS, with $O(n)$ reaching around 550 ms for 60 MB, attributed to its linear block-based encryption model. The Disjunctive Scheme exhibits the lowest encryption time ($\approx$ 550 ms for 60 MB) due to its simpler structure and lack of policy-driven overheads, despite also having $O(n^2)$ complexity.

(b) *Decryption Time (Figure 7(b)):* The decryption times mirror the encryption pattern, with PBCS again incurring the highest decryption cost due to the need for complex access control verifications and policy-based checks ($O(n^2)$). At 60 MB, decryption takes approximately 975 ms, mainly due to the polynomial complexity of reconstructing keys from collaborative shares θ and the overhead of policy checks. Yang et al. exhibit faster decryption due to its $O(n)$ complexity, aligned with its block-wise decryption method, peaking around 600 ms for larger files. The Disjunctive scheme, with its simpler $O(n^2)$ decryption model, performs faster than PBCS but remains slower than Yang et al., reaching 780 ms at maximum file size. This modest decryption overhead is the tradeoff for collaborative policy enforcement and stronger governance over co-owned data.

(c) *Deletion Time (Figure 7(c)):* PBCS offers an efficient, constant deletion time of 10 ms, independent of file size, by employing a key-based deletion mechanism that operates in $O(n)$ complexity. This involves invalidating keys through predefined polynomials (i.e., after deletion polynomial construction), ensuring rapid compliance with data erasure policies. In contrast, Yang et al. achieve $O(n \log n)$ complexity due to the Merkle tree structure required for deletions, with time increasing logarithmically with file size (e.g., 70 ms for 60 MB). Each deletion requires recalculating hashes along the Merkle tree, consuming more resources as the data size grows. While PBCS handles deletions more efficiently for large data volumes, Yang et al.'s scheme exhibits higher computational costs owing to the Merkle tree verification and block approach.

(d) *Deletion Validation Time within PBCS (Figure 7(d)):* This analysis confirms PBCS offers efficient deletion validation using a binary indicator for irreversible data erasure, with constant validation time independent of group count, underscoring low complexity. Deletion validation functions as a final check, leveraging predefined $E(x)$ and $D(x)$ polynomials, which bypasses the need for polynomial recalculations. As the number of co-owners grows, the time cost rises with the Lagrange

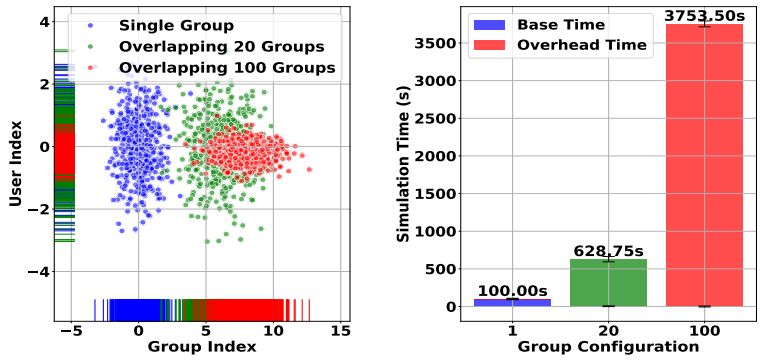


Fig. 8. Scalability in complex configurations: impact of overlapping co-owners across various groups.

polynomial's increasing degree, adding computational demands per user share. For $n = 5$ users, time cost is around 55 μs; for $n = 60$, it rises to 100 μs. This increase is due to the linear overhead in deletion validation as more co-owners are included.

Scalability (Figure 8). We tested PBCS under complex configurations with overlapping co-owners across multiple group structures, totaling 1,000 trials; we also evaluated **(i)** *partial adversarial entropy* by replacing a subset of public sources with predictable/constant values and **(ii)** a *CP-skips-decay* failure where the provider omits active-decay/validation, in both cases observing that validation rejects invalid/stale shares as intended. For comparative baselines (Figure 7), at 60 MB PBCS incurs ≈750 ms encryption and ≈975 ms decryption versus ≈550 ms/ ≈600 ms for Yang et al. and ≈550 ms/ ≈780 ms for the Disjunctive scheme, while PBCS deletion remains ≈10 ms versus ≈70 ms for Yang et al. In PBCS, computation is dominated by threshold-based key reconstruction and conjunctive validation, while communication is dominated by distributing shares and collecting votes/acks from participating co-owners. Thus, communication grows approximately linearly with the number of participating co-owners. In contrast, computation increases with the effective reconstruction threshold and is amplified by more conjunctive groups and higher overlap, rather than policy text length. ***(a)*** *Single Group (500 COs):* The system shows minimal overhead with 500 co-owners in a single group. The total processing time is around 100 seconds, indicating smooth scalability with relatively straightforward validation and key reconstruction. ***(b)*** *20 Groups (500 Overlapping COs):* Distributing the same 500 co-owners across 20 groups introduces moderate overhead due to collaborative validation and repeated group-level checks under overlap, with the total processing time rising to 628.75 seconds. Although scalability remains manageable, key recovery operations and validation checks start to introduce noticeable delays that could be reduced by streamlining these processes. ***(c)*** *100 Groups (500 Overlapping COs):* The system encounters significant performance bottlenecks when spreading 500 overlapping co-owners across 100 groups. As the number of groups grows, the interactions among overlapping co-owners become increasingly computationally intensive; the total processing time reaches approximately 3,753.50 seconds. This reflects the added complexity of managing large-scale conjunctive validation for key recovery, particularly under highly distributed configurations.

5.4 Limitations

Decay and Refresh Coordination: Reliability depends on synchronized share refresh and decay (Theorem 3). Poor synchronization in highly distributed systems may prematurely corrupt keys. In our design, refresh is permitted only before the policy expiration time t_{max}: if a refresh interval t

has passed and $t < t_{max}$, new points are generated; once $t \geq t_{max}$, periodic refresh stops and the points follow the decay factor λ (Definition 2), ensuring the key is forgotten. Thus, expiry/active decay takes precedence, and refresh cannot extend validity beyond t_{max}; any refresh event at/after t_{max} is ignored.

Large-scale deployment and $O(N^2)$ polynomial cost: Our evaluation in Section 5 focuses on per-object performance under hundreds of co-owners/groups and complex overlap scenarios. It does not benchmark cloud-scale workloads with millions of objects. The primary computational bottleneck is Lagrange-style polynomial construction/interpolation, which can incur $O(N^2)$ work per object/policy, where N is the number of participating points. At very large N or high policy churn, scaling requires engineering optimizations such as parallelizing independent group checks, batching validations/reconstructions, caching/reusing policy polynomials across many objects with identical policies, and adopting faster interpolation/evaluation techniques; with parallel processing, total time could theoretically approach the slowest-group cost rather than the sum across groups, while batching and caching reduce repeated overhead; exploring these optimizations at production scale is left for future work.

Conflict resolution and vote withdrawal. The current PBCS design does not define a vote-withdrawal or rollback procedure once a deletion decision is formed; after the deletion condition is satisfied and deletion is triggered, the resulting erasure is intended to be irreversible. Supporting "pending deletion" with explicit withdrawal/appeal windows (e.g., commit-after-delay) is an operational governance extension and is left for future work.

6 Related Work

Data Deletion Mechanisms. Yang et al. [79] proposed a blockchain-based scheme enabling publicly verifiable data deletion in untrustworthy cloud environments. Xiong et al. [77] tailored key derivation encryption for IoT flash memory hierarchies, enabling key deletion upon data expiration. Yu et al. [83] used **attribute-based encryption** (**ABE**) to ensure data deletion via attribute revocation, while Yang et al. [81] utilized counting Bloom filters for secure deletion. Kaushik and Gandhi [43] designed a capability-based system with assured deletion through user revocation. Ma et al. [48] introduced **secure data deletion with verification** (**SDVC**) using an attribute association tree for rapid revocation and re-encryption. Such and Criado [73] employed conflict resolution strategies based on shared data sensitivity. Enrico et al. [8] suggested fragment slicing with iterative re-encryption for efficient revocation. Filipe [11] developed Scramble, enabling social network users to encrypt, revoke, and delete data. Olteanu et al. [57] focused on consensual photo uploads by removing detectable faces. PBCS advances these approaches by providing an audience-based expiration and governance for collaborative co-owners.

Collaborative Access. Huang et al. [40] developed a **Lagrange interpolation-driven access control mechanism** (**LIDACM**) for securing **Personal Health Records** (**PHRs**) with randomly generated keys and a two-stage verification process. Xue et al. [78] proposed a CP-ABE scheme for public cloud storage, allowing users to combine attributes for collaborative access control. Carminati et al. [19] introduced collaborative security policies in online social networks, involving multiple users in access decisions based on relationships. Ilia et al. [42] presented SAMPAC as a privacy-preserving data-sharing system in which co-owners collaboratively set access controls, which are managed by a TTP (i.e., KMS). In He et al. [39], the authors propose an efficient CP-ABE scheme integrated with blockchain to support collaborative decryption. Using a linear secret-sharing scheme, the scheme reduces computational overhead and addresses key management issues through a multi-authorization model. Despite these advancements, existing models fall short in handling overlapping memberships and managing complex group dynamics in collaborative data access/deletion, a gap PBCS effectively addresses.

Table 3. Comparison with Previous Works [27] and [80]

Feature	Disjunctive [27]	Yang et al. [80]	PBCS
Model	Amortized	Amortized	Amortized
TTP	✗	✗	✗
Verifiability [70]	✗	$\mathcal{P}_{\mathrm{u}}$	$\mathcal{P}_{\mathrm{r}}$
Encrypt	$O(n^2)$	$O(n)$	$O(n^2)$
Decrypt	$O(n^2)$	$O(n)$	$O(n^2)$
Delete	✗	$O(n \log n)$	$O(n)$
Verify	✗	$O(n \log n)$	$O(n \log n)$
Access Control	Limited	✗	High
Dynamic Membership	✗	✗	✓
Collaborative Deletion	✗	✗	✓
Access and Deletion Policy	✗	✗	✓
Autonomy	Low	Low	High

(1) n: Number of data blocks, **(2)** $\mathcal{P}_{\mathrm{r}}$: Private verifiability, **(3)** $\mathcal{P}_{\mathrm{u}}$: Public verifiability.

Verifiable/Certified Deletion and Post-Quantum Variants. Recent work formalizes *(publicly) verifiable/certified deletion*, where a party outputs evidence that outsourced data is no longer retrievable. In the classical setting, generic compilers add publicly-verifiable deletion to standard primitives from minimal assumptions, and complementary systems work studies verifiable deletion for outsourced co-owned data, enabling any verifier to check deletion evidence [10, 26, 45]. In the quantum setting, certified deletion leverages quantum states so that producing a deletion certificate implies the plaintext is unrecoverable, even if keys are later revealed, and extends to quantum proofs of deletion with public verification in LWE-based constructions [17, 62]. Compared with these works, PBCS targets *co-ownership*: deletion is governed by multi-party consent (majority/veto) and enforced via *cryptographic erasure* (key irrecoverability under threshold/decay), with *private* validation for the owner/co-owners rather than public evidence of provider-side wiping.

Entropy Assessment. Because passive decay uses time-varying inputs, we connect to entropy assessment guidance and empirical analyses of candidate sources. NIST SP 800-90B defines tests and methodology for evaluating entropy sources used in random-bit generation [71], and prior work analyzes real signals under that framework (e.g., ECG-derived sources) [59]. In PBCS, entropy is used as a passive decay driver to avoid static shares and hinder snapshot retention, while hard guarantees rely on threshold secrecy/one-way recomputation and policy-triggered active decay; SP 800-90B-style assessment is therefore mainly a deployment aid for selecting or hardening internal sources.

Production Key Management and Ephemeral Keys (KMS). Operationally, PBCS aligns with standard envelope-encryption and key-lifecycle guidance (NIST SP 800-57) [9]. Cloud providers implement ephemeral **data-encryption keys** (**DEKs**) wrapped by long-lived **key-encryption keys** (**KEKs**) in managed KMS systems (e.g., AWS KMS GenerateDataKey / envelope encryption, Azure Key Vault wrap-key, and Google Cloud KMS envelope encryption) [3, 4, 38, 50]. PBCS complements these deployments by governing *when* a DEK remains reconstructable (and by whom) under co-owner policies and decay. At the same time, KMS provides hardened custody/auditing for PKI keys or wrapped share material, without affecting the policy-defined decay semantics.

Comparative Analysis of Schemes (Table 3). We compare the Disjunctive Scheme [27], Yang et al. [80], and PBCS across key metrics. All schemes adopt an amortized model, thereby avoiding

reliance on TTPs. PBCS excels in private verifiability, dynamic membership management, and collaborative deletion, ensuring co-owner autonomy and flexible access control. It is particularly suited to scenarios that demand stringent policy enforcement and audience-based data erasure. Yang et al.'s scheme supports public verifiability and strict compliance with forgetting regulations but lacks collaborative deletion and dynamic access control. The Disjunctive Scheme offers basic encryption/decryption with limited policy support, which is insufficient for complex collaborative environments. PBCS uniquely manages co-owner rights, dynamic memberships, and fine-grained policies, ensuring secure data forgetting.

7 Conclusion and Future Work

In response to the need for advanced data outsourcing in collaborative settings like literary collectives, this article introduces the *Policy-Based Conjunctive Scheme* (PBCS). The scheme enhances file sharing and conjunctive digital forgetting by enabling data owners to define policies for access and deletion, fostering active co-owner participation in data lifecycle management. PBCS integrates privacy and security features, including group categorization, membership validation, flexible access control, and reliable deletion mechanisms. Prototype evaluations demonstrate its effectiveness in decay analysis, performance evaluation, and formal validation, demonstrating its potential for secure cloud data management. Future work includes exploring multi-dimensional decay, Homomorphic Encryption, and quantum-resistant algorithms for provable data deletion.

References

[1] Shane Ahern, Dean Eckles, Nathaniel S. Good, Simon King, Mor Naaman, and Rahul Nair. 2007. Over-exposed? Privacy patterns and considerations in online and mobile photo sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, USA, 357–366.

[2] Hanaa Alshareef, Raúl Pardo, Gerardo Schneider, and Pablo Picazo-Sanchez. 2020. A collaborative access control framework for online social networks. *Journal of Logical and Algebraic Methods in Programming* 114 (2020), 100562.

[3] Amazon Web Services. 2025. AWS KMS API: GenerateDataKey. Retrieved from https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKey.html

[4] Amazon Web Services. 2025. AWS KMS cryptography: Envelope encryption. Retrieved from https://docs.aws.amazon.com/kms/latest/developerguide/kms-cryptography.html

[5] Ghous Amjad, Muhammad Shujaat Mirza, and Christina Pöpper. 2018. Forgetting with puzzles: Using cryptographic puzzles to support digital forgetting. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. ACM, USA, 342–353.

[6] Shashank Arora and Pradeep K. Atrey. 2021. Secure collaborative editing using secret sharing. In *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, IEEE, USA, 1–6.

[7] Oshrat Ayalon and Eran Toch. 2013. Retrospective privacy: Managing longitudinal privacy in online social networks. In *Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, UK, 1–13.

[8] Enrico Bacis, Sabrina De Capitani di Vimercati, Sara Foresti, Stefano Paraboschi, Marco Rosa, and Pierangela Samarati. 2016. Mix&Slice: Efficient access revocation in the cloud. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, Vienna, 217–228.

[9] Elaine Barker. 2020. *Recommendation for Key Management: Part 1 – General*. Technical Report SP 800-57 Part 1 Rev. 5. National Institute of Standards and Technology (NIST). Retrieved from https://csrc.nist.gov/pubs/sp/800/57/pt1/r5/final

[10] James Bartusek, Dakshita Khurana, Giulio Malavolta, Alexander Poremba, and Michael Walter. 2023. Weakening Assumptions for Publicly-Verifiable Deletion. Retrieved from https://arxiv.org/pdf/2304.09846 arXiv:2304.09846.

[11] Filipe Beato, Markulf Kohlweiss, and Karel Wouters. 2011. Scramble! your social network data. In *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, Springer, USA, 211–225.

[12] Mihir Bellare and Phillip Rogaway. 2005. *Introduction to Modern Cryptography*. UCSD CSE. Technical Report. 207 (2005), 207.

[13] Theo Bertram, Elie Bursztein, Stephanie Caro, Hubert Chao, Rutledge Chin Feman, Peter Fleischer, Albin Gustafsson, Jess Hemerly, Chris Hibbert, Luca Invernizzi, et al. 2019. Five years of the right to be forgotten. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 959–972.

[14] Martin Berz and Georg Hoffstätter. 1998. Computation and application of Taylor polynomials with interval remainder bounds. *Reliable Computing* 4, 1 (1998), 83–97.

[15] John Bethencourt, Amit Sahai, and Brent Waters. 2007. Ciphertext-policy Attribute-based Encryption. In *2007 IEEE Symposium on Security and Privacy (SP'07)*. IEEE, 321–334.
[16] Matt Bishop, Emily Rine Butler, Kevin Butler, Carrie Gates, and Steven Greenspan. 2013. Forgive and forget: return to obscurity. In *Proceedings of the 2013 New Security Paradigms Workshop*. ACM, USA, 1–10.
[17] Anne Broadbent and Rabib Islam. 2020. Quantum Encryption with Certified Deletion. In *Theory of Cryptography Conference (TCC)*. Springer, 92–122. Retrieved from https://dl.acm.org/doi/10.1007/978-3-030-64381-2_4
[18] Nairen Cao, Adam O'Neill, and Mohammad Zaheri. 2020. Toward RSA-OAEP without random oracles. In *IACR International Conference on Public-Key Cryptography*. Springer, 279–308.
[19] Barbara Carminati and Elena Ferrari. 2011. Collaborative access control in on-line social networks. In *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. IEEE, 231–240.
[20] Claude Castelluccia, Emiliano De Cristofaro, Aurélien Francillon, and Mohamed-Ali Kaafar. 2011. Ephpub: Toward robust ephemeral publishing. In *2011 19th IEEE International Conference on Network Protocols*. IEEE, Canada, 165–175.
[21] Ann Cavoukian et al. 2021. Privacy by design: The seven foundational principles. *IAPP Resource Center*, Retrieved from https://iapporg/resources/article/privacy-by-design-the-7-foundational-principles (2021).
[22] Chin-Chen Chang, Ngoc-Tu Huynh, and Hai-Duong Le. 2014. Lossless and unlimited multi-image sharing based on Chinese remainder theorem and Lagrange interpolation. *Signal Processing* 99 (2014), 159–170.
[23] Peter Cibik, Patrik Dobias, Sara Ricci, Jan Hajny, Lukas Malina, Petr Jedlicka, and David Smekal. 2024. Pushing AES-256-GCM to Limits: Design, Implementation and Real FPGA Tests. In *International Conference on Applied Cryptography and Network Security*. Springer, 303–318.
[24] Kovila P. L. Coopamootoo and Thomas Groß. 2017. Why privacy is all but forgotten. In *Proceedings on Privacy Enhancing Technologies* 2017, 4 (2017), 97–118.
[25] Joan Daemen. 2020. *The Design of Rijndael: The Advanced Encryption Standard (AES)*. Springer; 2nd edition, USA.
[26] Marwan Adnan Darwish, Evangelia Anna Markatou, and Georgios Smaragdakis. 2025. Provable co-owned data deletion with zero-residuals and verifiability in multi-cloud environment. In *Proceedings of the 18th European Workshop on Systems Security*. 77–83.
[27] Marwan Adnan Darwish and Georgios Smaragdakis. 2024. Disjunctive multi-level digital forgetting scheme. In *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 112–121.
[28] Marwan Adnan Darwish, Eiad Yafi, Abdullah H. Almasri, and Megat F. Zuhairi. 2018. Privacy and security of cloud computing: A comprehensive review of techniques and challenges. *International Journal of Engineering Trends and Technology* 7, 4.29 (2018), 239–246.
[29] Marwan Adnan Darwish and Apostolis Zarras. 2023. Digital forgetting using key decay. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. ACM, USA, 34–41.
[30] Biswajit Das and Dhritikesh Chakrabarty. 2016. Lagrange's interpolation formula: representation of numerical data by a polynomial curve. *International Journal of Mathematics Trend and Technology* 34, 2 (2016), 23–31.
[31] Emiliano De Cristofaro, Claudio Soriente, Gene Tsudik, and Andrew Williams. 2012. Hummingbird: Privacy at the time of twitter. In *2012 IEEE Symposium on Security and Privacy*. IEEE, 285–299.
[32] Adriana Deac et al. 2018. Regulation (eu) 2016/679 of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and the free movement of these data. *Perspectives of Law and Public Administration* 7, 2 (2018), 151–156.
[33] Amandeep Dhir, Puneet Kaur, Kirsti Lonka, and Marko Nieminen. 2016. Why do adolescents untag photos on Facebook? *Computers in Human Behavior* 55 (2016), 1106–1115.
[34] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. 2020. Formalizing data deletion in the context of the right to be forgotten. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 373–402.
[35] Roxana Geambasu, Tadayoshi Kohno, Arvind Krishnamurthy, Amit Levy, Henry M. Levy, Paul Gardner, and Vinnie Moscaritolo. 2011. New directions for self-destructing data. *University of Washington, Tech. Rep. UW-CSE-11-08-01* (2011).
[36] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. 2009. Vanish: Increasing Data Privacy with Self-Destructing Data. In *USENIX Security*, Vol. 316. USENIX, USA, 299–350.
[37] Henri Gilbert and Helena Handschuh. 2003. Security analysis of SHA-256 and sisters. In *International Workshop on Selected Areas in Cryptography*. Springer, 175–193.
[38] Google Cloud. 2025. Cloud KMS: Envelope encryption. Retrieved from https://docs.cloud.google.com/kms/docs/envelope-encryption
[39] Ying He, Haiyan Wang, Yuan Li, Ke Huang, Victor C. M. Leung, F. Richard Yu, and Zhong Ming. 2021. An efficient ciphertext-policy attribute-based encryption scheme supporting collaborative decryption with blockchain. *IEEE Internet of Things Journal* 9, 4 (2021), 2722–2733.

[40] Yin-Tzu Huang, Dai-Lun Chiang, Tzer-Shyong Chen, Sheng-De Wang, Fei-Pei Lai, and Yu-Da Lin. 2022. Lagrange interpolation-driven access control mechanism: Towards secure and privacy-preserving fusion of personal health records. *Knowledge-Based Systems* 236 (2022), 107679.
[41] François Hublet, David Basin, and Srđan Krstić. 2023. Enforcing the GDPR. In *European Symposium on Research in Computer Security*. Springer, 400–422.
[42] Panagiotis Ilia, Barbara Carminati, Elena Ferrari, Paraskevi Fragopoulou, and Sotiris Ioannidis. 2017. SAMPAC: Socially-aware collaborative multi-party access control. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy*. 71–82.
[43] Shweta Kaushik and Charu Gandhi. 2020. Capability based outsourced data access control with assured file deletion and efficient revocation with trust factor in cloud computing. *International Journal of Cloud Applications and Computing (IJCAC)* 10, 1 (2020), 64–84.
[44] Mohammad Taha Khan, Maria Hyun, Chris Kanich, and Blase Ur. 2018. Forgotten but not gone: Identifying the need for longitudinal data management in cloud storage. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.
[45] Fuyuki Kitagawa, Ryo Nishimaki, and Takashi Yamakawa. 2023. Publicly verifiable deletion from minimal assumptions. In *Theory of Cryptography Conference*. Springer, 228–245.
[46] Dexter Kozen and Marc Timme. 2007. *Indefinite Summation and the Kronecker Delta*. Technical Report. Cornell University. (2007).
[47] Bin Li, Yu Fu, and Kun Wang. 2022. A review on cloud data assured deletion. In *2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT)*. IEEE, 451–457.
[48] Jun Ma, Minshen Wang, Jinbo Xiong, and Yongjin Hu. 2021. CP-ABE-based secure and verifiable data deletion in cloud. *Security and Communication Networks* 2021 (2021), 1–14.
[49] Viktor Mayer-Schönberger. 2011. *Delete: The Virtue of Forgetting in the Digital Age*. Princeton University Press, USA.
[50] Microsoft. 2025. Azure Key Vault Keys: Wrap Key operation. Retrieved from https://learn.microsoft.com/en-us/rest/api/keyvault/keys/wrap-key/wrap-key
[51] Shujaat Mirza and Christina Pöpper. 2021. My past dictates my present: Relevance, exposure, and influence of longitudinal data on facebook. In *Workshop on Usable Security and Privacy (USEC)*.
[52] Paul V. Mockapetris. 1987. RFC 1035: Domain names—implementation and specification. *Internet Engineering Task Force (IETF)* (November 1987).
[53] Richard A. Mollin. 2002. *RSA and Public-Key Cryptography*. Chapman and Hall/CRC.
[54] Mainack Mondal, Günce Su Yilmaz, Noah Hirsch, Mohammad Taha Khan, Michael Tang, Christopher Tran, Chris Kanich, Blase Ur, and Elena Zheleva. 2019. Moving beyond set-it-and-forget-it privacy settings on social media. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 991–1008.
[55] Ambar Murillo, Andreas Kramm, Sebastian Schnorf, and Alexander De Luca. 2018. " If i press delete, it's gone"-user understanding of online data deletion and expiration. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS)*. 329–339.
[56] Farzad Nourmohammadzadeh Motlagh, Seyed Ali Alhosseini, Feng Cheng, and Christoph Meinel. 2023. An approach to multi-party privacy conflict resolution for co-owned images on content sharing platforms. In *Proceedings of the 2023 8th International Conference on Machine Learning Technologies*. 264–268.
[57] Alexandra-Mihaela Olteanu, Kévin Huguenin, Italo Dacosta, and J-P Hubaux. 2018. Consensual and privacy-preserving sharing of multi-subject and interdependent data. In *Proceedings of the 25th Network and Distributed System Security Symposium (NDSS)*. Internet Society, 1–16.
[58] Fernando Ortega, Raúl Lara-Cabrera, Ángel González-Prieto, and Jesús Bobadilla. 2021. Providing reliability in recommender systems through Bernoulli matrix factorization. *Information Sciences* 553 (2021), 110–128.
[59] Lara Ortiz-Martin, Pablo Picazo-Sanchez, and Pedro Peris-Lopez. 2020. Are the interpulse intervals of an ECG signal a good source of entropy? An in-depth entropy analysis based on NIST 800-90B recommendation. *Future Generation Computer Systems* 105 (2020), 346–360.
[60] Radia Perlman. 2005. *The Ephemerizer: Making Data Disappear*. Technical report. Sun Microsystems, (2005).
[61] Eugenia Politou, Efthimios Alepis, and Constantinos Patsakis. 2018. Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions. *Journal of Cybersecurity* 4, 1 (2018), tyy001.
[62] Alexander Poremba et al. 2022. Quantum Proofs of Deletion for Learning with Errors. arXiv:2203.01610. Retrieved from https://arxiv.org/abs/2203.01610
[63] DA Quadling. 1966. Lagrange's interpolation formula. *The Mathematical Gazette* 50, 374 (1966), 372–375.
[64] Kopo M. Ramokapane, Awais Rashid, and Jose M. Such. 2016. Assured deletion in the cloud: Requirements, challenges and future directions. In *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*. 97–108.
[65] Kopo Marvin Ramokapane, Awais Rashid, and Jose Miguel Such. 2017. "I feel stupid I can't delete...": A study of users' cloud deletion practices and coping strategies. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS)*. 241–256.

[66] Sirke Reimann and Markus Dürmuth. 2012. Timed revocation of user data: Long expiration times from existing infrastructure. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. 65–74.
[67] Theodor Schnitzler, Shujaat Mirza, Markus Dürmuth, and Christina Pöpper. 2021. SOK: Managing longitudinal privacy of publicly shared personal online data. *Proceedings on Privacy Enhancing Technologies* 2021, 1 (2021), 229–249.
[68] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
[69] Raksha Sharma. 2023. Cloud Storage Market - Global Industry Analysis. Retrieved from https://growthmarketreports.com/report/cloud-storage-market-global-industry-analysis Accessed: April 20, 2024.
[70] Victor Shoup and Nigel P. Smart. 2024. Lightweight asynchronous verifiable secret sharing with optimal resilience. *Journal of Cryptology* 37, 3 (2024), 27.
[71] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry McKay, Mary Baish, and Michael Boyle. 2018. *Recommendation for the Entropy Sources Used for Random Bit Generation*. Technical Report SP 800-90B. National Institute of Standards and Technology (NIST). Retrieved from https://csrc.nist.gov/pubs/sp/800/90/b/final
[72] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review* 31, 4 (2001), 149–160.
[73] Jose M. Such and Natalia Criado. 2016. Resolving multi-party privacy conflicts in social media. *IEEE Transactions on Knowledge and Data Engineering* 28, 7 (2016), 1851–1863.
[74] Willy Susilo, Peng Jiang, Jianchang Lai, Fuchun Guo, Guomin Yang, and Robert H. Deng. 2021. Sanitizable access control system for secure cloud storage against malicious data publishers. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2021), 2138–2148.
[75] European Union. 2016. *Regulation on the Protection of Natural Persons with Regard to the Processing of Personal Data and on the Free Movement of Such Data, and Repealing Directive 95/46/EC (Data Protection Directive)*. Retrieved from https://en.wikipedia.org/wiki/General_Data_Protection_Regulation
[76] Gregor Wegberg, Hubert Ritzdorf, and Srdjan Capkun. 2017. *Multi-User Secure Deletion on Agnostic Cloud Storage*. Technical Report. ETH Zurich.
[77] Jinbo Xiong, Lei Chen, Md Zakirul Alam Bhuiyan, Chunjie Cao, Minshen Wang, Entao Luo, and Ximeng Liu. 2020. A secure data deletion scheme for IoT devices through key derivation encryption and data analysis. *Future Generation Computer Systems* 111 (2020), 741–753.
[78] Yingjie Xue, Kaiping Xue, Na Gai, Jianan Hong, David S. L. Wei, and Peilin Hong. 2019. An attribute-based controlled collaborative access control scheme for public cloud storage. *IEEE Transactions on Information Forensics and Security* 14, 11 (2019), 2927–2942.
[79] Changsong Yang, Xiaofeng Chen, and Yang Xiang. 2018. Blockchain-based publicly verifiable data deletion scheme for cloud storage. *Journal of Network and Computer Applications* 103 (2018), 185–193.
[80] Changsong Yang, Yueling Liu, Feng Zhao, and Shubin Zhang. 2022. Provable data deletion from efficient data integrity auditing and insertion in cloud storage. *Computer Standards & Interfaces* 82 (2022), 103629.
[81] Changsong Yang, Xiaoling Tao, Feng Zhao, and Yong Wang. 2020. Secure data transfer and deletion from counting bloom filter in cloud computing. *Chinese Journal of Electronics* 29, 2 (2020), 273–280.
[82] Sheng-Cheng Yeh, Ming-Yang Su, Hui-Hui Chen, and Chun-Yuen Lin. 2013. An efficient and secure approach for a cloud collaborative editing. *Journal of Network and Computer Applications* 36, 6 (2013), 1632–1641.
[83] Yong Yu, Liang Xue, Yannan Li, Xiaojiang Du, Mohsen Guizani, and Bo Yang. 2018. Assured data deletion with fine-grained access control for fog-based industrial applications. *IEEE Transactions on Industrial Informatics* 14, 10 (2018), 4538–4547.
[84] Lingfang Zeng, Zhan Shi, Shengjie Xu, and Dan Feng. 2010. SafeVanish: An improved data self-destruction for protecting data privacy. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. IEEE, 521–528.
[85] Zhaoyun Zhang, Qitong Wang, and Zhi Zhang. 2021. Harmonic vector error analysis based on lagrange interpolation. *IEEE Access* 9 (2021), 57464–57474.
[86] Lei Zhou, Anmin Fu, Yi Mu, Huaqun Wang, Shui Yu, and Yinxia Sun. 2021. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. *Information Sciences* 545 (2021), 254–276.

Appendices

A Preliminary Definitions

Definition 7. A symmetric encryption scheme consists of the following components:
Key Generation: A probabilistic algorithm $\text{KeyGen}(\kappa) \rightarrow k_s$ that, given a security parameter κ, generates a symmetric key k_s. This step is based on Definition 1.
Encryption: A deterministic function $\text{Enc}_{k_s} : \{0,1\}^* \rightarrow \{0,1\}^*$ that takes a plaintext m and outputs a ciphertext c.
Decryption: A deterministic function $\text{Dec}_{k_s} : \{0,1\}^* \rightarrow \{0,1\}^*$ that takes a ciphertext c and outputs the plaintext m. Formally, $c = \text{Enc}_{k_s}(m) \quad \text{and} \quad m = \text{Dec}_{k_s}(c)$, such that $\text{Dec}_{k_s}(\text{Enc}_{k_s}(m)) = m$.

Definition 8. An asymmetric encryption scheme consists of the following components:
Key Generation: A probabilistic algorithm $\text{KeyGen}(\kappa) \rightarrow (k_{pub}, k_{pri})$ that, given a security parameter κ, generates a public key k_{pub} and a private key k_{pri}.
Encryption: A deterministic function $\text{Enc}_{k_{pub}} : \{0,1\}^* \rightarrow \{0,1\}^*$ that takes a plaintext m and outputs a ciphertext c.
Decryption: A deterministic function $\text{Dec}_{k_{pri}} : \{0,1\}^* \rightarrow \{0,1\}^*$ that takes a ciphertext c and outputs the plaintext m. Formally, $c = \text{Enc}_{k_{pub}}(m)$ and $m = \text{Dec}_{k_{pri}}(c)$, such that $\text{Dec}_{k_{pri}}(\text{Enc}_{k_{pub}}(m)) = m$.

Definition 9. Cryptographic Hash Function $H : \{0,1\}^* \rightarrow \{0,1\}^n$ maps an input x of arbitrary length to a fixed-size string h of n bits. Formally, $h = H(x)$, where h is the hash value.

B Polynomial Transformation Analysis (Figure 9)

Polynomial transformations play a pivotal role in enhancing the security features of cryptographic systems such as (PBCS). These transformations involve modifying polynomial coefficients through specific algebraic operations, which are integral to the system's ability to resist algebraic attacks and ensure secure key evolution.

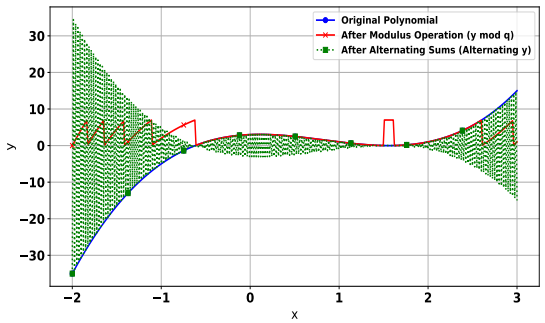


Fig. 9. Polynomial transformation: modulus and alternating sums functions.

B.1 Modulus Operations

The application of modulus operations with a prime number, typically denoted as q, to the coefficients of a polynomial $P(x) = \sum_{i=0}^{n} a_i x^i$ results in a transformed polynomial $P'(x) = \sum_{i=0}^{n} (a_i \mod q) x^i$. This operation ensures that all polynomial coefficients are reduced to a finite field $\mathbb{F}_q$,

which is crucial for maintaining polynomial behavior within computationally manageable bounds, crucial for both security and performance.

Example: Given a polynomial $P(x) = 3x^2 + 5x + 9$, and a prime $q = 5$, the modulus operation transforms $P(x)$ to $P'(x) = 3x^2 + 0x + 4$.

B.2 Alternating Sums

Alternating sums involve modifying the polynomial's coefficients by adding and subtracting adjacent coefficients. For a polynomial $P(x)$, this results in a new polynomial $P''(x) = a_0 - a_1x + a_2x^2 - a_3x^3 + \ldots$. This transformation can significantly alter the polynomial's zero points and overall algebraic structure, increasing the complexity of solving it algebraically.

Example: Continuing with $P'(x) = 3x^2 + 0x + 4$, applying alternating sums would result in $P''(x) = 4 - 0x + 3x^2$.

B.3 Cryptographic Implications

These transformations directly impact the system's resilience against algebraic attacks, such as attempts to derive private keys from public information or to break the system's encryption. By operating in a finite field and increasing the polynomial's algebraic complexity:

— The modulus operation confines the coefficients within a predictable range, preventing overflow and reducing susceptibility to timing attacks.
— Alternating sums increase the polynomial's algebraic irregularity, complicating potential factorization or root-finding algorithms that adversaries might employ.

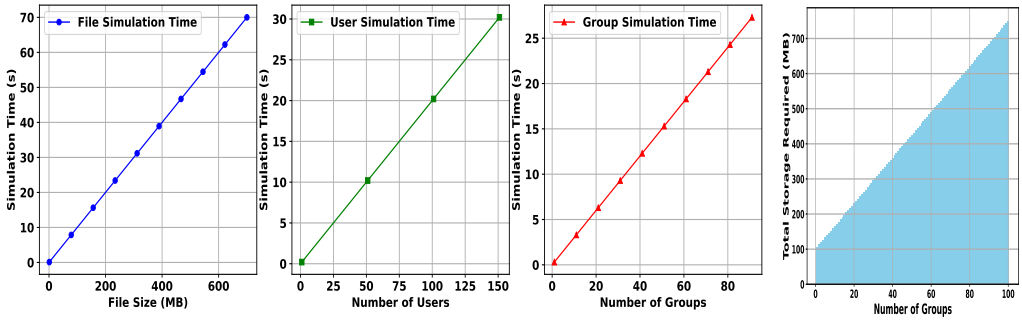
In conclusion, carefully applying these polynomial transformations in PBCS supports encryption mechanisms. It ensures that cryptographic keys remain secure throughout their lifecycle, adhering to principles such as the right to be forgotten, as mandated by regulations such as the GDPR.

C Sensitivity and Scalability Analysis Experiments

Decay Factor Impact on Key Effectiveness. The runtime statistics for varying decay factors and key effectiveness levels, from 70% to 20%, are detailed in Table 4. This table covers statistics for different decay models, including linear, exponential, and sigmoidal decay. The table highlights minimum, maximum, average, and standard deviation in runtime (in hours) across these effectiveness levels. As observed, linear decay gradually declines, while exponential decay accelerates the key decay more sharply. Sigmoidal decay provides a unique *S*-shaped pattern, starting with slower decay and then accelerating rapidly before stabilizing, which may align well with adaptive security needs.

Linear Scalability. In the scalability assessment of the conjunctive scheme, we evaluated three pivotal dimensions to reflect the scheme's adaptability to increasing operational demands (See Figure 10(a)). *(a) Scale of File per Group:* Empirical data reveals a linear increase in operational time from nearly zero for small files to about 60 seconds for 600 MB files, demonstrating predictable and manageable increments essential for diverse data volumes ($O(N)$ where N is the file size in MB). *(b) Users per Group:* Operation time increases proportionally with the number of users, from negligible for a single user to approximately 30 seconds for 150 users, ensuring scalability and support for an expanding user base ($O(N)$ where N is the number of users). *(c) Number of Groups:* Operation time correlates linearly with the number of groups, increasing from nearly zero for one group to about 25 seconds for 80 groups. This consistency is vital for maintaining access control and security in growing environments ($O(N)$ where N is the number of groups).

Storage Space Required. This metric evaluates the digital storage needed for encrypted data, control mechanisms, and logs crucial for monitoring and validating deletion processes, which



(a) Scalability Trends Study: File Size / Number of Users / Number of Groups Over Simulation Time.

(b) Total Storage Across Different Groups.

Fig. 10. Linear scalability trends and total storage used.

Table 4. Run-Time Statistics and Various Decay Factors Influencing Key Effectiveness $K(t)$ Levels

$K(t)$	**70%**	**60%**	**50%**	**40%**	**30%**	**20%**
Linear Decay ($\lambda = 0.020\,\mathbf{hr}^{-1}$)						
Min (hrs)	1.2395	2.5637	3.9254	5.2194	6.7432	8.4571
Max (hrs)	1.9564	3.7159	5.0192	6.7048	8.2431	10.2784
Avg (hrs)	1.5979	3.1398	4.4723	5.9621	7.4932	9.3678
St.Dev (hrs)	0.2926	0.4703	0.4465	0.6064	0.6123	0.7435
# Tests	5000	5000	1000	500	200	50
Exponential Decay ($\lambda = 0.1\,\mathbf{hr}^{-1}$)						
Min (hrs)	0.6213	1.2117	1.7104	2.8473	4.5193	6.8123
Max (hrs)	1.6742	2.4912	3.1987	3.9647	5.7891	8.9452
Avg (hrs)	1.1478	1.8515	2.4546	3.4060	5.1542	7.8788
St.Dev (hrs)	0.4298	0.5223	0.6075	0.4561	0.5183	0.8707
# Tests	5000	5000	1000	500	200	50
Sigmoidal Decay ($s = 10\,\mathbf{hr}^{-1}$)						
Min (hrs)	2.1952	3.0195	3.9451	4.7893	6.3429	8.5723
Max (hrs)	2.9918	4.2393	5.6247	6.8963	8.1952	11.2571
Avg (hrs)	2.5935	3.6294	4.7849	5.8428	7.2691	9.9147
St.Dev (hrs)	0.3252	0.4979	0.6856	0.8601	0.7561	1.0960
# Tests	5000	5000	1000	500	200	50

are key for scalability and cost-effectiveness in cloud environments. Our analysis quantified the storage overhead introduced by the conjunctive scheme, essential for assessing scalability and cost-efficiency. As shown in Figure 10(b), starting with a 100 MB baseline file, the storage requirement increases proportionally with the number of user groups, adding 2 MB per group for security metadata and an additional 1 MB per group for deletion proofs. The total storage, S, as a function of group count, N, is $S(N) = 3N + 100$ (in MB), confirming the scheme's linear scalability and providing a basis for cost estimation in real deployments.

ALGORITHM 10: Threshold Adaptation

Input: data_points, initial threshold θ_init, new threshold θ_new
Output: $Updated_data_points$

$L_x \leftarrow$ ConstructLagrangePolynomial($data_points$)
Generate $\theta_new - 1$ random data_points $(x_i, y_i) \in \mathbb{F}_q$
seed $\leftarrow$ EvaluateAtZero(L_x)
$Updated_data_points \leftarrow$ Generated points+$(0, seed)$
return $Updated_data_points$

ALGORITHM 11: Periodic Share Renewal

Input: data_points, refresh time t, expiration time t_max
Output: New_shares

$L_x \leftarrow$ ConstructLagrangePolynomial($data_points$)
// Every t timesteps
if t_{max} *time has passed* **then**
    return 0
end
$New_shares \leftarrow |data_points|$ random points from L_x
return New_shares

ALGORITHM 12: Group-Specific Public-Key Encryption for Polynomial Coefficients

Input: coefficients, publicKey
Output: Encrypted coefficients

return $\{\text{AsymEnc}(c_i, publicKey)\}_{c_i \in coefficients}$

ALGORITHM 13: Group-Specific Public-Key Decryption for Polynomial Coefficients

Input: encrypted_coefficients, privateKey
Output: Decrypted coefficients

return $\{\text{AsymDec}(c_i, privateKey)\}_{c_i \in coefficients}$

D Additional Algorithms in the Study

This section presents the algorithms used during the upload/download phase of PBCS, with a focus on access control mechanisms.

- **Algorithm 10**: Re-shares the same secret under a new reconstruction threshold.
- **Algorithm 11**: Periodically refreshes shares by sampling new points from the existing polynomial until expiration.
- **Algorithm 12**: Generates asymmetric encryption keys.
- **Algorithm 13**: Reconstructs them for decryption.
- **Algorithms 15 and 14**: Reconstruct the symmetric key for decryption, either singular or collaborative.
- **Algorithm 16**: Defines the participation matrix, determining group membership and co-owner overlap.
- **Algorithm 17**: Implements group-based access control using Lagrange polynomials to ensure collaborative access.

ALGORITHM 14: KSS.ReconstructKey

Input: datapoints *data_points*
Output: Key k

$L_x \leftarrow$ ConstructLagrangePolynomial(*data_points*)
seed $\leftarrow$ EvaluateAtZero(L_x)
key $\leftarrow$ H(seed)
return key k

ALGORITHM 15: Symmetric Key Reconstruction Type

Input: Participation matrix m, datapoints *Shares*, θ
Output: Symmetric Key k

Type $\leftarrow$ *singular*
if *there exists a user i, such that ParticipationBased_Matrix_Approach*$[m, i] = 1$ **then**
  Type $\leftarrow$ *collab*
end
if *there exist fewer than* θ *unique x values in Shares* **then**
  return *Error: "Please provide at least* θ *unique data points."*
end
return KSS.ReconstructKey(*Shares*) (See Algorithm 14)

ALGORITHM 16: Participation-Based Matrix Approach

Input: Membership M, User index i
Output: Participation decision (0 or 1)

participation_count $\leftarrow$ Sum of elements in $M[i]$
if *participation_count* ≥ 2 **then**
  return 1
end
return 0

ALGORITHM 17: Group-Based Access Control via Lagrange Polynomial

Input: $GroupIDs, GroupSignatures, Votes_{\text{acc}}, nonce$
Output: Access decision (0 or 1)

Function `VerifyAccess`$(GroupID, GroupSignature \oplus nonce[128])$**:**
  $H \leftarrow$ cryptographic hash function
  $Y \leftarrow$ empty array of size $|GroupIDs|$
  $L_{access} \leftarrow 0$
  $input_{ver} \leftarrow$ concatenate$(GroupID, GroupSignature)$
  $\hat{y}_{ver} \leftarrow H(input_{ver})$
  for $i = 1$ *to* $|GroupIDs|$ **do**
    $input \leftarrow$ concatenate$(GroupIDs[i], GroupSignatures[i])$
    $Y[i] \leftarrow H(input)$ $basis \leftarrow 1$
    for $j = 1$ *to* $|GroupIDs|$ **do**
      if $j \neq i$ **then**
        $basis \leftarrow basis \times \frac{x_{ver} - GroupIDs[j]}{GroupIDs[i] - GroupIDs[j]} \mod q$
      end
    end
    $L_{access} \leftarrow L_{access} + Y[i] \times basis \mod q$
  end
  return $(L_{access} \equiv \hat{y}_{ver} \mod q)$
foreach $group \in Votes_{acc}$ **do**
  $GroupID, GroupSignature \leftarrow group$
  AccessDecision $\leftarrow$ `VerifyAccess`$(GroupID, GroupSignature)$
  if ***AccessDecision*** *== 1* **then**
    return 1
  end
end
return 0

Received 30 March 2025; revised 11 February 2026; accepted 14 March 2026