The Right to Be Forgotten

Reinforcing Digital Data Forgetting in Cloud Storage

Darwish Khabbaz, M.

DOI
10.4233/uuid:fc651ca5-a759-43cd-b937-9d4287b98020

Publication date
2025

Document Version
Final published version

Citation (APA)
Darwish Khabbaz, M. (2025). *The Right to Be Forgotten: Reinforcing Digital Data Forgetting in Cloud Storage*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:fc651ca5-a759-43cd-b937-9d4287b98020

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

The Right to Be Forgotten

Reinforcing Digital Data Forgetting in Cloud Storage

Doctor of Philosophy

M-Marwan Darwish Khabbaz

Propositions

accompanying the dissertation

The Right to Be Forgotten

Reinforcing Digital Data Forgetting in Cloud Storage

by

M-Marwan DARWISH KHABBAZ

1. Retrospective privacy in cryptographic forgetting requires irreversible key destruction without external dependency (*Chapter 2*).
2. Without audience-specific expiration, digital forgetting fails to reflect the realities of contextual privacy (*Chapter 3*).
3. Collaborative data ownership demands shared control over both access and deletion to uphold privacy rights (*Chapter 4*).
4. Provable deletion in multi-cloud settings demands cryptographic enforcement, not provider trust (*Chapter 5*).
5. People forget, but the internet remembers everything.
6. Technology evolves faster than the ethics that govern it.
7. Language models compress collective memory, not understanding.
8. Long-term success demands prioritizing perseverance over talent.
9. A dream is a rebellion against the world as it is.
10. Peace may nurture intellect, but hardship forges strength.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors prof. dr. G. Smaragdakis, prof. dr. M. Conti and dr. E.A. Markatou.

The Right to Be Forgotten

Reinforcing Digital Data Forgetting in Cloud Storage

The Right to Be Forgotten

Reinforcing Digital Data Forgetting in Cloud Storage

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus,
Prof. dr. ir. T.H.J.J. van der Hagen,
chair of the Board for Doctorates
to be defended publicly on
Thursday 4 December 2025 at 12:30 o'clock

by

M-Marwan DARWISH KHABBAZ

Master of Information Technology, Kuala Lumpur University, Malaysia
born in Aleppo, Syria

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. G. Smaragdakis	Delft University of Technology, *promotor*
Prof. dr. M. Conti	Delft University of Technology, *promotor*
Dr. E.A. Markatou	Delft University of Technology, *copromotor*
Independent members:	
Prof. dr. ir. R.L. Lagendijk	Delft University of Technology
Prof. dr. H. Haddadi	Imperial College London, UK
Prof. dr. N. Laoutaris	IMDEA Networks Institute, Spain
Dr. F.S. Gürses	Delft University of Technology
Dr. G. Acar	Radboud University, Netherlands
Prof. dr. F.A. Kuipers	Delft University of Technology, *reserve member*

Keywords: Digital Forgetting, Key Decay, Audience-Based Expiration, Co-Owned Data Deletion, Verifiable Multi-Cloud Deletion

Printed by: ProefschriftMaken

Cover by: Haya Darwish Khabbaz

ISBN 978-94-6534-009-8

An electronic copy of this dissertation is available at
https://repository.tudelft.nl/.

"*If you venture in pursuit of glory, do not be satisfied with anything less than the stars.*"

— Al-Mutanabbī

CONTENTS

Summary xi

Samenvatting xiii

1 Introduction 1
- 1.1 Motivations for Digital Forgetting in Cloud Storage 3
  - 1.1.1 Data Obsolescence: The Digital Dustbin Dilemma . . 3
  - 1.1.2 User Autonomy: Empowering Data Owners 4
  - 1.1.3 Security Vulnerabilities: Minimizing Exposure 4
- 1.2 Digital Forgetting Definition 5
- 1.3 Regulatory Frameworks Governing Data Deletion 7
- 1.4 Ethical and Societal Considerations 8
- 1.5 Adversarial Models and Challenges in Digital Forgetting . . . 9
  - 1.5.1 Adversarial Models in Digital Forgetting 10
  - 1.5.2 Technical Challenges in Digital Forgetting 11
- 1.6 Problem Statement 13
  - 1.6.1 P1: Insufficient Retrospective Privacy Protection . . . 13
  - 1.6.2 P2: Lack of Flexible Audience-Specific Expiration Controls 14
  - 1.6.3 P3: Challenges in Secure Collaborative and Democratic Deletion 14
  - 1.6.4 P4: Inadequate Verifiability in Multi-Cloud Data Deletion 15
- 1.7 Research Questions 15
- 1.8 Contributions of the Thesis 16

2 Retrospective Privacy- Securing Ephemeral Keys 19
- 2.1 Introduction 20
- 2.2 Threat Model 22
- 2.3 High-Level Idea 22
  - 2.3.1 Security Goals 22
  - 2.3.2 Abstract Architecture 23
- 2.4 Scheme Description 24
  - 2.4.1 Key Generation Phase 24
  - 2.4.2 Encryption Phase 26
  - 2.4.3 Reconstruction Phase 26
  - 2.4.4 Decay Phase 27
- 2.5 Evaluation 28
  - 2.5.1 Randomization 28

2.5.2 Decay Rate . . . 29
2.5.3 Computational Complexity . . . 35
2.5.4 Interpolation Efficiency Under Entropy Decay . . . 35
2.5.5 Security Analysis . . . 38
2.5.6 Limitations . . . 39
2.6 Related Work . . . 40
2.7 Conclusion . . . 41

3 Audience-Based Expiration 43
3.1 Introduction . . . 44
3.2 Concept . . . 46
3.2.1 Terminology . . . 46
3.2.2 Security Goals . . . 46
3.2.3 Overall Architecture . . . 47
3.3 Scheme Description . . . 48
3.3.1 Background on Key Decay Scheme . . . 48
3.3.2 Multi-Level Forgetting Structure . . . 49
3.3.3 Smart Contracts . . . 52
3.3.4 Expiration and Group Factors . . . 54
3.4 System Design and Architecture . . . 55
3.5 Results . . . 58
3.5.1 Decay Sensitivity . . . 58
3.5.2 Computational Complexity . . . 63
3.5.3 Security Analysis . . . 64
3.6 Related Work . . . 65
3.7 Conclusion . . . 67

4 Data Co-Ownership Digital Forgetting 69
4.1 Introduction . . . 70
4.2 Design Goals . . . 72
4.2.1 Problem Statement . . . 72
4.2.2 System Actors and Threat Models . . . 73
4.3 Concepts . . . 73
4.3.1 Definitions . . . 73
4.3.2 Security Definitions . . . 76
4.3.3 Terminology . . . 77
4.3.4 System Architecture . . . 78
4.4 Scheme Description . . . 80
4.4.1 Conjunctive Key Decay Scheme . . . 80
4.4.2 A Policy-Based Conjunctive Scheme (PBCS) . . . 82
4.4.3 Formal Analysis . . . 101
4.4.4 Adversarial Models Analysis . . . 103
4.5 Results and Insights . . . 104
4.5.1 IAM-Based Policies for PBCS Protocol . . . 105
4.5.2 Evaluation Metrics . . . 108
4.5.3 Experimental Results . . . 108

4.6 Related Work . . . 115
4.6.1 Data Deletion Mechanisms. . . . 115
4.6.2 Collaborative Access. . . . 117
4.6.3 Comparative Analysis of Schemes. . . . 118
4.7 Conclusion And Future Work . . . 118

5 Verifiable Co-Owned Data Deletion in Multi-Cloud Environments 119
5.1 Introduction . . . 120
5.2 Design Goals . . . 123
5.2.1 Asynchronous and Secure Communication . . . 123
5.2.2 Threat Model . . . 123
5.2.3 Security Model . . . 124
5.3 Preliminaries . . . 125
5.3.1 Bounded Merkle Hash Trees (BMHT) . . . 125
5.3.2 Global Merkle Forest (GMF) . . . 127
5.4 System Architecture . . . 127
5.5 Scheme Description . . . 129
5.5.1 Initialization . . . 129
5.5.2 Verification via BMHT and Proofs . . . 131
5.5.3 Data Sharing, Secure Access, Zero-Residuals Overwriting, and Global Consistency . . . 133
5.6 Formal Analysis . . . 137
5.7 Results and Discussion . . . 140
5.7.1 Computational Complexity . . . 140
5.7.2 Experimental Results . . . 141
5.8 Related Work . . . 148
5.9 Conclusion . . . 150

6 Discussion 151
6.1 Retrospective Privacy . . . 152
6.2 Audience-Based Expiration . . . 153
6.3 Data Co-Ownership Digital Forgetting . . . 153
6.4 Verifiable Co-Owned Data Deletion in Multi-Cloud Environments . . . 154
6.5 Societal and Ethical Implications of Digital Forgetting . . . 155
6.5.1 Use Case 1: Cloud-Based Services and Autonomy . . . 155
6.5.2 Use Case 2: Collaborative Digital Authorship and Fairness . . . 156
6.5.3 Use Case 3: Healthcare Data Management and Patient Privacy . . . 156
6.6 Limitations and Future Work . . . 157
6.6.1 Key Decay . . . 157
6.6.2 Disjunctive Multi-Level Digital Forgetting Scheme . . . 158
6.6.3 Conjunctive Scheme for Digital Forgetting of Co-Owned Data . . . 158

6.6.4 Provable Deletion of Co-Owned Data in Cloud Environment 159
6.6.5 Towards Machine Unlearning in Privacy-Preserving Systems 159

Acknowledgements **181**

Curriculum Vitæ **185**

List of Publications **187**

SUMMARY

The exponential growth of digital data has reshaped the global landscape of information management, posing urgent security, compliance, and sustainability challenges. Cloud computing offers scalable, ubiquitous storage but raises critical concerns about data lifecycle governance, especially the irreversible deletion of data that has outlived its purpose. Digital forgetting becomes the art of purposeful disappearance in a cloud that remembers everything. The thesis addresses this pressing question: **Can cloud-stored data ever be truly forgotten?**

To answer this, the thesis presents four interrelated contributions to reinforce digital data forgetting in cloud storage: advancing privacy-preserving forgetting, enabling audience-specific expiration control, supporting collaborative deletion for co-owned data, and ensuring verifiable erasure in untrusted multi-cloud environments.

To address retrospective privacy, we propose Key Decay, a cryptographic scheme where encryption keys degrade irreversibly over time, eliminating reliance on ephemeral storage and enhancing data expiration guarantees.

To support audience-specific data expiration, we propose a Disjunctive Multi-Level Forgetting Scheme that enables distinct user groups to access the same data under tailored validity periods. Smart contracts and decay sensitivity tuning enforce flexible governance across hierarchical access levels.

To manage co-owned data deletion, we introduce a Policy-Based Conjunctive Scheme that accommodates overlapping group memberships and collaborative decision-making. It applies conjunctive thresholds and verifiable key decay that comply with secure forgetting under the EU General Data Protection Regulation (GDPR) Right to Be Forgotten in real-world multi-stakeholder settings.

To ensure verifiable deletion under Byzantine infrastructure, we design a Verifiable Deletion Framework for Multi-Cloud Environments, combining Hardware Security Modules, Secure Enclaves, and dual-layer Merkle hashing to produce cryptographic proofs of deletion across providers both locally and globally.

Together, these contributions form a unified, privacy-preserving framework for managing cloud data from creation to irreversible deletion, reinforcing secure digital forgetting and regulatory compliance.

SAMENVATTING

De exponentiële groei van digitale data heeft het mondiale landschap van informatiebeheer ingrijpend veranderd en brengt aanzienlijke uitdagingen met zich mee op het gebied van beveiliging, naleving van regelgeving en duurzaamheid. Cloudcomputing biedt schaalbare en alomtegenwoordige opslagmogelijkheden, maar roept tegelijkertijd ernstige zorgen op over de governance van de levenscyclus van data, vooral met betrekking tot het onomkeerbaar verwijderen van gegevens die hun doel hebben gediend. Digitaal vergeten wordt de kunst van het doelgericht verdwijnen uit een cloud die zich alles herinnert. Dit proefschrift behandelt de centrale vraag: **Kan data opgeslagen in de cloud ooit echt vergeten worden?**

Om deze vraag te beantwoorden, presenteert dit proefschrift vier onderling verbonden bijdragen die digitaal vergeten in cloudopslag versterken: het bevorderen van privacy-behoudend vergeten, het ondersteunen van audience-specifieke vervaltijden, het mogelijk maken van gezamenlijke verwijdering van gegevens bij co-eigendom, en het verzekeren van controleerbare verwijdering in niet-vertrouwde multi-cloudomgevingen.

Om retroactieve privacy mogelijk te maken stellen we sleutelverval voor, een cryptografisch schema waarin sleutels op een gegeven moment onomkeerbaar vervallen. Dit voorkomt de nood om te vertrouwen op tijdelijke opslag en verzekert het verval van data.

Om een audience-specifieke vervaltijd te ondersteunen, introduceren we een Disjunctief Meerlagig Vergetingsschema. Hiermee krijgen verschillende gebruikersgroepen toegang tot dezelfde gegevens, elk met een eigen, op maat gemaakte geldigheidsduur. Smart contracts en decay-sensitiviteitstuning handhaven flexibel beleid over hiërarchische toegangsniveaus.

Om gezamenlijke verwijdering van gedeelde gegevens mogelijk te maken, introduceren we een Beleidsgebaseerd Conjunctief Schema dat rekening houdt met overlappende groepslidmaatschappen en collaboratieve besluitvorming. Het past conjunctieve drempels en verifieerbare sleutelverval toe die voldoen aan veilig vergeten onder de EU General Data Protection Regulation (GDPR) Right to Be Forgotten in praktische multi-stakeholderomgevingen.

Om controleerbare verwijdering binnen Byzantijnse infrastructuren te waarborgen, ontwerpen we een Controleerbaar Verwijderingsframework voor multi-cloudomgevingen. Dit framework combineert Hardware Security Modules, Secure Enclaves en dubbellaagse Merkle-hashing om cryp-

tografisch bewijs van verwijdering te leveren, zowel lokaal als wereldwijd.

Al deze bijdragen samen vormen een verenigd privacy-behoudend framework voor het beheer van data in de cloud, van aanmaking tot verwijdering. Zo versterkt dit framework het veilig digitaal vergeten en de naleving van de regelgeving.

1

INTRODUCTION

Data generation and consumption have exploded in the 21st century, fueled by the fast digitalization of all aspects of industry and life. Global data generation and consumption have grown unprecedentedly, reaching 64.2 zettabytes in 2020. Projections by IDC and Statista [1, 2] estimate this figure will exceed 180 zettabytes by the end of 2025 and rise to 394 zettabytes by 2028. The increasing reliance on digital infrastructure across sectors such as personal communication, healthcare, finance, and enterprise operations drives this monumental growth. As data accumulates at scale, its effective management, security, and long-term preservation have become critical concerns for individuals and organizations.

Cloud computing emerged as a vital solution to such challenges, offering scalable, flexible storage solutions and shying away from on-premises infrastructure investments [3–5]. Cloud services enable users to access, process, and retrieve data from virtually anywhere [6]. This fosters operational efficiency and collaborative workflows without the need for physical presence. Additionally, modern cloud platforms integrate advanced security measures, including encryption, redundancy mechanisms, and access control protocols, to safeguard data from unauthorized access, cyber threats, and accidental loss [7]. The growing dependence on cloud computing is evident in its widespread adoption across industries. According to *Edge Delta*, a cloud observability platform, approximately 94% of businesses worldwide have adopted cloud services [8]. While adoption is widespread, market forecasts project that the global cloud computing industry will surpass $1 trillion by 2028, reflecting its growing role in digital transformation and scalable data management.

From creation to final forgetting, data in cloud environments follows a defined lifecycle that includes multiple stages of processing, access, and eventual deletion [9–12]. After being created by human activity, Internet of Things sensors, automated systems, or business apps, data is saved

and arranged in cloud repositories—typically provisioned and paid for by the organization or individual that owns the data—for effective indexing and retrieval. Ownership generally resides with the entity that generated the data, which also defines who is authorized to access it. Under established access control policies, authorized individuals may access shared data across many platforms. Businesses and individuals can extract value and draw significant conclusions by converting raw data into actionable insights through processing and analysis. Organizations often retain data for operational continuity, compliance, or historical reference, following legal or regulatory requirements. Eventually, data reaches the forgetting phase, where it must be securely removed from cloud systems to prevent unauthorized access or misuse.

Despite the many merits of the cloud environment, ensuring data's secure and irreversible deletion remains one of the most pressing challenges [13–17]. Although deletion marks the intended final stage of the data lifecycle, distributed cloud environments complicate this process due to replication, backups, and caching [18, 19].

The concern becomes greater when data involves personal or sensitive content, such as health records, user profiles, or financial logs. A central question drives this thesis:

Is the data genuinely and permanently forgotten when a data owner requests deletion?

As more data is created and stored, from personally identifiable information (PII) and financial records to confidential business documents, the sensitivity and stakes surrounding that data grow. This makes the ability to securely and permanently delete information not just a technical requirement but a core concern for privacy protection, regulatory compliance, and the long-term trustworthiness of cloud systems.

This chapter introduces digital forgetting, outlining its significance and associated challenges. Section 1.1 presents the motivation, highlighting the risks of persistent digital data and the necessity for enforceable deletion mechanisms. Section 1.2 defines digital forgetting, while Section 1.3 examines relevant legal and regulatory frameworks. Ethical and societal considerations are discussed in Section 1.4, addressing the balance between privacy and accountability. Section 1.5 identifies key adversarial and technical challenges affecting digital forgetting. The core problem statement and research questions are detailed in sections 1.6 and 1.7, respectively. Finally, Section 1.8 provides an overview of the thesis structure and contributions.

1.1. MOTIVATIONS FOR DIGITAL FORGETTING IN CLOUD STORAGE

The cloud storage environment has transformed how data is stored, accessed, and managed, offering scalable infrastructure and near-ubiquitous availability on demand. However, this convenience comes at a cost: digital records, unlike physical ones, do not degrade naturally over time. Data can persist indefinitely across replicated and archived cloud systems unless deliberately removed. This enduring presence accumulates outdated, redundant, or sensitive information, raising serious concerns for privacy, security, and regulatory compliance.

The need for digital forgetting stems from deeper concerns about individual privacy, data autonomy, and long-term security. This section outlines three key motivations: the unchecked persistence of obsolete data, the loss of user control over deletion, and the security risks that arise from prolonged exposure of sensitive information.

1.1.1. DATA OBSOLESCENCE: THE DIGITAL DUSTBIN DILEMMA

Not all data is meant to last forever. Information that was once useful, such as outdated financial records, obsolete project files, or temporary backups, often loses its relevance over time [20]. In cloud environments, the continued presence of such data is often the result of automated replication, long-term archival mechanisms, and the absence of mandatory expiration policies.

This persistent availability is not merely inefficient—it poses real risks to personal privacy. Obsolete data may resurface unexpectedly, be taken out of context, or be used in ways never intended by the individual it concerns. This can result in reputational harm, misinformation, or privacy violations, particularly when the information is sensitive or personally identifiable. This reality undermines the fundamental *rights to privacy* and *digital autonomy* and directly contradicts the principle of *data minimization*, which requires that personal data be collected and retained only as long as necessary for a specified purpose [21]. The call for digital forgetting in the cloud is therefore driven by the need to protect individual rights, not just to remove obsolete data but to ensure that people can reclaim control over what remains accessible about them online.

To address this, digital systems must adopt structured deletion mechanisms that recognize when data has outlived its purpose and ensure it is no longer accessible, regardless of how many copies exist across cloud infrastructures.

1.1.2. USER AUTONOMY: EMPOWERING DATA OWNERS

Deleting a file from their device or online account is expected to be a definitive action for many users. However, in cloud environments, deletion is rarely straightforward. Cloud providers prioritize redundancy and fault tolerance, replicating data across multiple servers and regions to prevent loss from accidental deletion or failure. While this enhances durability, it significantly weakens a user's ability to control the lifecycle of their content.

The situation becomes even more complex in collaborative environments. Once a file is shared, other participants can duplicate, modify, or redistribute it. These additional copies may persist independently of the original, often beyond the awareness or control of the initial uploader. As a result, a single deletion request is insufficient to ensure comprehensive data removal. This *"copy proliferation"* issue is one of the most complex problems in cloud data governance. Without mechanisms to track, manage, and enforce deletion across all derivatives of shared content, digital ownership becomes fragmented and uncertain. Users cannot be assured that their information is gone, even when local deletion appears successful [22].

To uphold the principle of user autonomy, deletion must go beyond interface-level actions—it must address the structural complexity of cloud replication and shared access. Enforceable lifecycle policies are essential to empower users with meaningful control over their digital footprint.

1.1.3. SECURITY VULNERABILITIES: MINIMIZING EXPOSURE

The failure to securely delete outdated or unneeded data significantly increases the risk of long-term security breaches [23]. In cloud environments, retained data—even if encrypted—can become a target for adversaries who seek to exploit its eventual decryptability. One growing concern is the *"harvest now, decrypt later"* model [24], where attackers collect encrypted data today intending to break its encryption in the future using advanced or quantum-capable techniques.

Even when data is encrypted today, retaining it indefinitely gives future attackers more time to compromise the keys or break the underlying cryptographic primitives. This delayed risk undermines assumptions of forward secrecy and elevates the importance of timely and irreversible deletion. Beyond targeted attacks, excessive retention also creates internal blind spots. Organizations managing large volumes of data often struggle to maintain visibility into what should be retained or removed. Forgotten records, unsecured backups, or misclassified datasets have all contributed to large-scale breaches in the past.

Minimizing exposure requires more than access control—it demands proactive deletion mechanisms that enforce the principle of data

minimization. Only actively used and justifiably retained data should persist; all else should be securely erased to reduce the window of vulnerability and limit the potential for future exploitation.

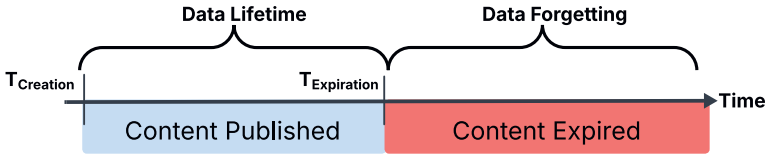


Figure 1.1: Conceptual representation of digital forgetting. The data lifecycle begins at T_{Creation} with the content being published and remaining accessible until its designated expiration time $T_{\text{Expiration}}$. After expiration, the content enters the digital forgetting phase, signifying its removal or inaccessibility over time.

1.2. DIGITAL FORGETTING DEFINITION

Information inevitably loses relevance and significance within a specific digital context over time. Digital data forgetting is the deliberate and structured process of erasing, concealing, or limiting access to previously stored or shared digital information [25]. Unlike natural forgetting in human memory, which passively occurs as experiences fade or lose importance, digital traces often persist unless actively removed. While some digital content may gradually decay due to obsolescence or system turnover, this passive fading is typically slow, inconsistent, and insufficient for meeting modern privacy expectations or regulatory demands. Therefore, digital forgetting requires deliberate intervention to effectively reduce individuals' digital footprint. It encompasses methods that render certain personal information inaccessible, obsolete, or significantly less retrievable without necessitating complete erasure or historical revision, as illustrated in Figure 1.1.

In the digital era, the ability to forget has become as important as the ability to remember [26, 27]. The persistent availability of digital data poses significant challenges to privacy, autonomy, and the human right to evolve and reinvent oneself. While digital records can degrade over time due to system turnover or obsolescence, their ease of duplication and replication means that they often persist far beyond their original, context-specific purpose, such as fulfilling a transaction, supporting communication, or ensuring compliance. This prolonged retention increases the risk of privacy breaches, identity theft, and the misuse of outdated information, particularly when taken out of context. Digital forgetting thus emerges as a necessary counterbalance to the long-term memory of online platforms and databases, ensuring that

information no longer serving its legitimate function does not remain indefinitely accessible.

Digital forgetting is not merely about pressing "delete"—it involves implementing enforceable, privacy-preserving mechanisms to ensure that data is no longer accessible, usable, or identifiable once it has outlived its purpose [28, 29]. Individuals have the right to take control of their digital presence, deciding what stays and what disappears. At the same time, organizations must balance regulatory compliance with operational efficiency, ensuring data is retained only for as long as it serves a legitimate purpose. This complexity is amplified in cloud computing, where files, messages, and records are duplicated across multiple servers—often spread across different countries, each governed by unique legal frameworks [25].

Unlike absolute physical deletion, digital forgetting is a broader, layered process that can be implemented through technological, legal, ethical, and societal strategies. While deletion refers to the technical removal of data, such as erasing bits from storage, digital forgetting encompasses irretrievability, policy enforcement, and trace elimination. In cloud environments, where deletion alone is often insufficient due to hidden replicas, system logs, or backups, actual forgetting demands enforceability, cryptographic assurance, and systemic control. Approaches to digital forgetting include access control, cryptographic mechanisms, deception, and even linguistic strategies. Common technical methods include cryptographic erasure (e.g., deleting encryption keys), unlinking files at the system level, secure data overwriting, degaussing magnetic media, and physically destroying storage devices.

In addition to technical and policy-based mechanisms, digital forgetting strategies can also be conceptually classified into four categories [30]: (1) controlling dissemination (restricting access to data), (2) hiding (making data difficult to locate or interpret), (3) inconveniencing the interpreter (leveraging leaked data to mislead or slow adversaries), and (4) establishing effectiveness metrics (quantifying the success of forgetting mechanisms).

From an ethical perspective, digital forgetting supports the idea that past actions should not indefinitely define individuals. Limiting the long-term visibility of outdated or no longer relevant data helps ensure a fair balance between accountability and personal development [31, 32].

Legally and socially, digital forgetting aligns closely with initiatives such as the European Union's "**Right to Be Forgotten**" [33], reflecting broader cultural and legal movements toward empowering individuals with control over their digital persona. However, this must be balanced against societal interests in transparency, historical accuracy, and freedom of information, thus preventing misuse that could equate digital forgetting with censorship or rewriting history.

The critical question becomes how data is deleted and whether it can be forgotten in a digital world where backups, logs, and distributed systems allow information to linger indefinitely. Addressing this challenge requires more than just new policies—it demands significant technical innovation, cryptographic guarantees, and a fundamental shift in how we perceive digital control and the information lifecycle.

1.3. REGULATORY FRAMEWORKS GOVERNING DATA DELETION

Several regulations have been enacted to address the challenges associated with data retention and deletion. These legal frameworks establish guidelines for organizations to manage, process, and delete personal data while ensuring compliance with privacy and security standards.

GENERAL DATA PROTECTION REGULATION (GDPR)

The *General Data Protection Regulation* (GDPR), implemented by the European Union in 2018, introduced comprehensive data protection principles. *Article 17* enshrines the *Right to Erasure* or *Right to Be Forgotten (RTBF)*, granting individuals the ability to request the deletion of their data.

> **GDPR Article 17 [34]:** *"The data subject shall have the right to obtain from the controller the erasure of personal data concerning him or her without undue delay."*

This regulation mandates that organizations comply with deletion requests promptly, provided that legal and operational conditions are met, reinforcing individual control over personal data.

CALIFORNIA CONSUMER PRIVACY ACT (CCPA)

The *California Consumer Privacy Act* (CCPA), enacted in 2018, grants California residents specific rights regarding their personal information. Among its key provisions is the right to request data deletion.

> **CCPA [35, 36]:** *"A consumer shall have the right to request that a business delete any personal information about the consumer which the business has collected from the consumer."*

This empowers consumers to demand the deletion of their data, requiring businesses to establish effective data management and deletion protocols to ensure compliance.

EUROPEAN HEALTH DATA SPACE (EHDS)

The *European Health Data Space* (EHDS) is a framework designed to facilitate the ethical use of health data for research and policy-making while maintaining strict privacy protections. It is the first common EU data space [37] dedicated to a specific sector as part of the European strategy for data. A key principle of the EHDS is the responsible secondary use of health data.

> **EHDS [38]:** *"Providing a trustworthy and efficient set-up for the use of health data for research, innovation, policy-making, and regulatory activities (secondary use of data)."*

This regulation aims to balance the advantages of leveraging health data for innovation with the necessity of protecting individual privacy.

HEALTH INSURANCE PORTABILITY & ACCOUNTABILITY ACT (HIPAA)

In the United States, the *Health Insurance Portability and Accountability Act* (HIPAA) establishes national standards for safeguarding health information. It mandates that covered entities implement strict measures for ensuring the confidentiality, integrity, and proper disposal of electronic health records.

> **HIPAA [39, 40]:** *"Covered entities must implement policies and procedures to ensure the proper disposal of protected health information (PHI) to prevent unauthorized access."*

HIPAA enforces strict requirements for data retention and disposal, ensuring that health records are permanently removed when no longer needed to prevent unauthorized access and misuse.

1.4. ETHICAL AND SOCIETAL CONSIDERATIONS

While regulatory frameworks establish the legal obligations surrounding data deletion, ethical considerations go beyond compliance, addressing the broader societal implications of data persistence. Responsible digital information management is a matter of regulatory adherence and maintaining trust, preventing harm, and respecting individual autonomy [41, 42].

RESPECTING INDIVIDUAL PRIVACY AND AUTONOMY

Ensuring that personal data is not retained longer than necessary aligns with ethical principles that emphasize individual privacy and autonomy. When organizations and service providers continue to store outdated information, individuals lose control over their digital footprint. Ethical data management necessitates that users retain the ability to determine when and how their data is removed, preventing prolonged exposure of sensitive information.

MITIGATING POTENTIAL HARM

The indefinite retention of digital data increases the risk of unintended exposure, data breaches, and misuse. Improperly managed data can be exploited by malicious actors, leading to financial fraud, identity theft, or reputational damage. Secure data deletion practices help mitigate these risks by ensuring that obsolete or unnecessary information is securely and permanently erased, reducing opportunities for unauthorized access and exploitation.

ENHANCING TRUST AND TRANSPARENCY

Transparent and enforceable data deletion policies are essential for fostering trust between individuals and organizations. When users are confident that their data is handled responsibly and that deletion requests are honored, they are more likely to engage with digital services without fear of losing control over their personal information. Establishing transparent, verifiable deletion processes strengthens the relationship between data controllers and users, promoting ethical data stewardship in a rapidly evolving digital landscape.

1.5. ADVERSARIAL MODELS AND CHALLENGES IN DIGITAL FORGETTING

Digital forgetting mechanisms must tackle various technical challenges while mitigating adversarial threats that seek to bypass expiration policies, recover deleted data, or manipulate access conditions, as illustrated in Figure 1.2 [43]. These adversaries exploit system vulnerabilities, retention loopholes, and cryptographic weaknesses to achieve their objectives. Below, we analyze different adversarial models and their associated technical challenges, providing a foundation for defining the explicit research problems addressed in this thesis.

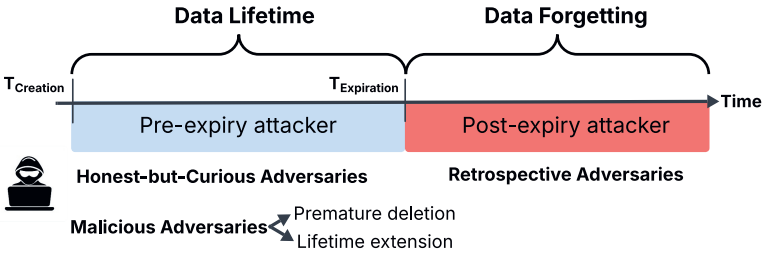


Figure 1.2: Considered attacker model: Before expiration ($T_{\text{Expiration}}$), adversaries include *honest-but-curious* observers and *interfering* adversaries manipulating data lifespan. After expiration, *retrospective* attackers attempt data recovery.

1.5.1. ADVERSARIAL MODELS IN DIGITAL FORGETTING

RETROSPECTIVE ADVERSARIES

These adversaries aim to recover expired encryption keys or deleted data by exploiting historical system states, such as archived storage snapshots, backups, or forensic traces [44]. Their focus is on accessing information *after* it has expired or been revoked. Likely actors include state-level agencies, forensic analysts, individuals with access to decommissioned hardware, and cloud providers with privileged access to system-level artifacts. Their capabilities range from access to archived or residual data to advanced cryptanalysis techniques, making them a critical threat to ephemeral key storage and long-term privacy guarantees.

A notable example is the *"harvest now, decrypt later"* model [24], where encrypted data is collected and stored today to decrypt it in the future as computational power advances.

HONEST-BUT-CURIOUS ADVERSARIES

These adversaries do not interfere with system operations but seek to gain insights from data before its expiration [43]. Typical actors include cloud providers, infrastructure administrators, or storage service operators with legitimate access, but may use that access to inspect or infer sensitive information passively. Their capabilities include reading metadata, logging access patterns, analyzing usage behavior, and generating analytical models without violating formal deletion procedures. This model emphasizes the necessity of cryptographic protections and auditability to counter silent observation.

MALICIOUS ADVERSARIES

These adversaries actively interfere with data lifecycle policies to undermine digital forgetting [45]. They may delay expiration to retain sensitive data longer or delete content prematurely to obstruct access for legitimate users. Actors may include compromised providers, colluding group members, or malicious cloud tenants. Their capabilities include direct manipulation of access control, alteration of timestamps, suppression of deletion triggers, and coordination with other compromised entities to disrupt the enforcement of deletion policies, particularly in multi-user or shared environments.

MALICIOUS CO-OWNERS IN MULTI-USER ENVIRONMENTS

Shared data environments introduce adversarial scenarios where co-owners exploit loopholes in deletion policies to retain or expose data without consent [43]. For instance, in collaborative cloud storage, shared files involve multiple contributors with differing deletion preferences. Some co-owners may refuse to comply with removal requests, while others manipulate access privileges to retain or expose sensitive content. This adversarial model highlights the need for structured governance mechanisms ensuring fair and enforceable deletion policies.

These adversarial models underscore the necessity of robust digital forgetting mechanisms that address privacy risks, ensure verifiable deletion, and support collaborative governance. Next, we define the technical challenges that arise from these threats.

1.5.2. TECHNICAL CHALLENGES IN DIGITAL FORGETTING

Digital forgetting mechanisms must resolve several key technical challenges to ensure secure, enforceable, and verifiable data deletion. These challenges stem from adversarial threats, rigid expiration policies, multi-owner governance complexities, and verifiable data removal requirements [46, 47]. Formally articulating these technical challenges provides a structured foundation for understanding the limitations addressed in this research.

EPHEMERAL KEY EXPOSURE

A significant technical challenge arises from adversaries exploiting system snapshots and historical data to reconstruct expired ephemeral keys, undermining privacy assurances post-deletion [48]. Let $K(t)$ be an ephemeral key active at time t (i.e., the key remains valid only for a limited time [49]) associated with a secure key management function G. The security condition requires that after expiration at t_{expire}, the probability of reconstructing $K(t)$ from historical state S remains negligible:

1

$$P(\text{Retrieve}(K(t)) \mid S) \leq \epsilon_{\text{key}}, \quad \forall t \geq t_{\text{expire}}, \quad \epsilon_{\text{key}} \to 0 \text{ as } t \to \infty.$$

For an encrypted data block D with ciphertext $C = E_{K(t)}(D)$, the adversary's advantage $\text{Adv}_{\mathcal{A}}(t)$ in decrypting C after expiration must also remain bounded:

$$\text{Adv}_{\mathcal{A}}(t) = P(\mathcal{A}(C) = D) \leq \epsilon_{\text{key}}.$$

This technical requirement transitions directly into the problem of insufficient retrospective privacy protection of the ephemeral keys.

INFLEXIBLE AUDIENCE-SPECIFIC EXPIRATION

Current expiration mechanisms impose uniform, static expiration policies [50]. Consider shared data object D accessible by distinct audience groups $A_1, A_2, \ldots, A_n$. Traditional mechanisms provide a fixed expiration time t_{expire} regardless of audience differences:

$$f_{\text{expire}}(D) = t_{\text{expire}}, \quad \forall A_i.$$

An effective mechanism must allow dynamic expiration thresholds t_i per audience A_i, thereby enforcing flexible and individualized policies:

$$P(\text{Access}(D) \mid A_i, t) = \begin{cases} 1, & \text{if } t < t_i \\ 0, & \text{if } t \geq t_i \end{cases}, \quad \forall A_i.$$

COLLABORATIVE DELETION GOVERNANCE

In multi-owner scenarios, conflicting deletion preferences among stakeholders pose a significant challenge [51, 52]. Let D be a data object co-owned by users $o_1, o_2, \ldots, o_m$ with respective deletion preferences d_i. Traditional deletion mechanisms inadequately resolve conflicting deletion requests. A robust governance model must implement a structured decision-making process f, ensuring fairness and compliance with defined policies:

$$\mathcal{D} = f(d_1, d_2, \ldots, d_m), \quad \text{where } f \text{ resolves conflicts democratically.}$$

VERIFIABLE DELETION AND PROOF-OF-ERASURE

A fundamental technical challenge is ensuring permanent and provable deletion in distributed cloud storage systems $\mathcal{S}$ [53, 54]. Existing solutions lack cryptographic proofs verifying successful deletion across inter-providers. Given a deletion request R for data D, a verification mechanism $V(R)$ must cryptographically guarantee:

$$V(R) = \begin{cases} \text{True} & \text{if } D \notin \mathcal{S}, \\ \text{False} & \text{otherwise.} \end{cases}$$

To achieve this, a cryptographic commitment $\mathcal{C}(D)$ ensures proof-of-erasure guarantees. Let $H(D)$ be a secure cryptographic hash function that uniquely represents the data:

$$H : D \to \{0,1\}^{\lambda}.$$

where λ is the bit length of the hash output. Using this, the proof-of-erasure must satisfy:

$$H(D) = \mathcal{C}(D), \quad \text{and upon deletion, } P(\mathcal{C}(D) \in \mathcal{S}) \to 0.$$

This ensures that once data is deleted, no retrievable commitment remains, preventing any future reconstruction.

These adversarial models and technical challenges set the stage for clearly articulating the explicit research problems addressed in this thesis, as detailed in the following problem statement.

1.6. PROBLEM STATEMENT

This thesis tackles the challenge of achieving secure and verifiable digital forgetting in cloud environments. It targets four technical challenges mentioned above: irreversible key expiration, audience-specific data expiration, collaborative deletion in shared ownership, and verifiable erasure across multi-cloud systems. To address these, it proposes cryptographic and policy-based solutions that enhance user autonomy and ensure privacy compliance.

1.6.1. P1: INSUFFICIENT RETROSPECTIVE PRIVACY PROTECTION

Traditional digital forgetting approaches often fail to ensure irreversible data deletion, allowing residual or retrievable data to persist post-deletion attempt. One of the primary concerns arises from the ability of retrospective adversaries to recover expired encryption keys through caching mechanisms, system snapshots, or forensic analysis. These attacks enable the reconstruction of previously encrypted data, even after it is marked for deletion.

Consider D as the data object intended for deletion and K as the set of cryptographic keys associated with D. In cloud environments, where data is redundantly stored and backed up, an attacker can exploit system snapshots to retrieve K even after its expiration. This allows them to decrypt D, effectively bypassing digital forgetting mechanisms. Current methods fail to ensure that the probability of reconstructing the data after deletion, $P(D \mid K_{deleted})$, approaches zero. Formally:

$$P(D_{recovery} \mid K_{deleted}, S) \gg 0,$$

where S represents historical snapshots or cache remnants.

An ideal deletion mechanism must enforce strict key expiration policies and cryptographic guarantees to ensure that once K is deleted, retrieval of D becomes infeasible. This condition is expressed as:

$$\lim_{t \to \tau_{exp}} P(D_{recovery}(t) \mid S) = 0,$$

where τ_{exp} represents the specified expiration time.

1.6.2. P2: LACK OF FLEXIBLE AUDIENCE-SPECIFIC EXPIRATION CONTROLS

Existing digital forgetting mechanisms typically impose uniform expiration policies, disregarding the diverse expiration needs of different stakeholders for the same data object. Suppose $U = \{u_1, u_2, \ldots, u_n\}$ represents distinct users or groups accessing data D. Traditional methods provide a static expiration T_{exp}, such that:

$$\forall u_i \in U, \quad T_{exp}(u_i) = constant.$$

This inflexibility does not support individualized data lifecycles. Therefore, an advanced approach must enforce individualized expiration conditions:

$$\forall u_i \in U, \exists\ T_{exp}(u_i) : T_{exp}(u_i) \neq T_{exp}(u_j), \quad \text{for } i \neq j.$$

Failure to support audience-specific expiration results in premature data deletion for specific stakeholders or unnecessary retention beyond the intended duration.

1.6.3. P3: CHALLENGES IN SECURE COLLABORATIVE AND DEMOCRATIC DELETION

In multi-owner environments, enforcing fair deletion policies is complex due to users' simultaneous participation in multiple groups, each with different expiration policies for the same shared content. Current models fail to balance audience-specific expiration and co-owner involvement for a collaborative environment, leading to governance inconsistencies.

Let $O = \{o_1, o_2, \ldots, o_m\}$ be co-owners of a shared data object D, and $G = \{G_1, G_2, \ldots, G_n\}$ be the groups with independent expiration policies $T_{exp}(G_k)$. A co-owner o_i in multiple groups faces conflicts when deletion rules differ:

$$T_{del}(o_i) = f(T_{exp}(G_1), T_{exp}(G_2), \ldots, T_{exp}(G_n)),$$

where f determines the effective deletion time. Prior models, such as Disjunctive Forgetting and rigidly segmented users, prevent overlapping memberships and fail to account for dynamic participation. This creates key challenges:

1. **Conflicting expiration policies:** A deletion request from one group may contradict another group's retention policy, leading to governance deadlocks.
2. **Dynamic group membership:** Users frequently join and leave groups, requiring an adaptive expiration model:

$$T_{exp}(o_i, t) = \begin{cases} T_{exp}(G_k) & \text{if } o_i \in G_k, \\ \infty & \text{otherwise.} \end{cases}$$

3. **Lack of co-owner autonomy:** If D is encrypted with a group key K_G, co-owners depend on centralized deletion triggers, limiting self-sovereign control.
4. **Lack of verifiable deletion:** A deletion request should ensure data removal across all cloud replicas via a cryptographic proof π:

$$\forall G_k, \quad \text{VerifyDeletion}(D, G_k, \pi) = \{\text{True, False}\}.$$

1.6.4. P4: Inadequate Verifiability in Multi-Cloud Data Deletion

Multi-cloud environments amplify the complexity of verifiable data deletion due to distributed replication and redundancy across multiple cloud providers [55]. Let $C = \{c_1, c_2, \ldots, c_k\}$ represent multiple cloud service providers that store copies of data D. Existing deletion mechanisms typically lack cryptographic assurances, denoted by π, to confirm successful deletion across all providers:

$$\exists c_j \in C, \quad \text{VerifyDeletion}(D, c_j, \pi) \rightarrow \text{ambiguous.}$$

Robust deletion verification methods must fulfill:

$$\forall c_j \in C, \quad \text{VerifyDeletion}(D, c_j, \pi) = \{\text{True, False}\}, \quad \text{with certainty.}$$

Cloud providers may retain residual data copies without cryptographic proof-of-erasure mechanisms, violating privacy regulations.

1.7. Research Questions

The research questions posed in this thesis are as follows:

RQ1: How can ephemeral keys be protected against retrospective adversaries?

RQ2: How can audience-based expiration be achieved across heterogeneous groups with disjunctive access conditions?

RQ3: How can we correctly ensure digital data forgetting for co-owned data?

RQ4: How can secure deletion of co-owned data be provably verified across multi-cloud infrastructures?

1.8. CONTRIBUTIONS OF THE THESIS

This thesis comprises multiple self-contained chapters, each based on individual research publications, with minor modifications where necessary. While the introduction and discussion provide overarching context, the technical chapters can be read independently, collectively contributing to the theme of secure and enforceable digital forgetting. Due to the integration of multiple publications, some background and problem definitions may appear across chapters. Each chapter contextualizes its study within the broader research framework to ensure coherence. The dissertation's structural outline is presented below.

CHAPTER 2: RETROSPECTIVE PRIVACY- SECURING EPHEMERAL KEYS

This chapter introduces an ephemeral key framework to address **RQ1** to mitigate retrospective adversaries attempting to recover expired encryption keys without relying on ephemeral storage. As digital storage grows, ensuring that keys remain accessible only during their intended lifespan is critical for privacy preservation. This work formalizes *key decay*, a technique that gradually and irreversibly corrupts encryption keys over time, rendering past decryption infeasible. The proposed model integrates randomness and cryptographic obfuscation to estimate key expiration, mitigating snapshot-based adversarial reconstruction. The framework is validated through security proofs and computational complexity analysis.

CHAPTER 3: AUDIENCE-BASED EXPIRATION

In response to **RQ2**, this chapter introduces a *Disjunctive Multi-Level Forgetting Scheme* to enforce audience-based expiration while

maintaining access control flexibility. Existing digital forgetting models impose uniform expiration, failing to accommodate diverse user requirements. The proposed approach assigns hierarchical expiration levels, ensuring selective deletion for specific audiences while allowing retention for others until their designated expiration periods elapse. This model leverages cryptographic key decay to automate expiration and smart contracts to enforce secure, verifiable deletions. The framework is evaluated regarding decay sensitivity, computational efficiency, and security guarantees.

CHAPTER 4: DATA CO-OWNERSHIP DIGITAL FORGETTING

To address **RQ3**, this chapter presents a *Policy-Based Conjunctive Scheme (PBCS)* to enable collaborative and policy-driven data deletion in co-ownership settings. As shared digital data becomes prevalent, uniform deletion models fail to accommodate co-owner preferences. The proposed framework allows owners to securely upload content while allowing co-owners to make deletion decisions via a conjunctive voting mechanism. Cryptographic techniques, including Lagrange interpolation-based decay, ensure data irretrievability based on pre-defined policies and collective agreement. The model complies with the European Union's *General Data Protection Regulation (GDPR)* and enhances structured governance in collaborative environments.

CHAPTER 5: VERIFIABLE CO-OWNED DATA DELETION IN MULTI-CLOUD ENVIRONMENTS

In response to **RQ4**, this chapter introduces a *Provable Co-Owned Data Deletion* framework for malicious multi-cloud environments, ensuring secure deletion with cryptographic verification. Existing deletion models primarily focus on individual ownership, often lacking transparency and auditability in multi-owner contexts. The proposed approach enables data owners to outsource encrypted content while allowing co-owners to perform computations securely within their respective cloud providers, ensuring that sensitive data never leaves the cloud. The system integrates *Hardware Security Modules (HSMs)* for cryptographic key management and *Secure Enclaves* for privacy-preserving computations. Deletion is enforced using *zero-residuals permuted overwriting* to ensure irreversible data removal. Verifiability is achieved through a two-tier system: *Bounded Merkle Hash Trees (BMHT)* ensure local integrity, while a *Global Merkle Forest (GMF)* aggregates BMHT roots for cross-cloud deletion verification. The framework is formally analyzed for security, experimentally validated, and assessed for performance and cost efficiency.

CHAPTER 6: DISCUSSION

This chapter reflects on the research questions addressed throughout the thesis, discusses their limitations, outlines possible directions for future work, and highlights the findings' broader societal and ethical implications.

2

RETROSPECTIVE PRIVACY-SECURING EPHEMERAL KEYS

During the recent development of information technology and the prevalent breakthroughs of its services, more digital data tend to be readily stored online. Although the massive advantages, there is a pivotal necessity for curating digital data forgetting. Online content can pose perilous threats in terms of privacy and security that may hinder the right to be forgotten, encompassed by the GDPR act, since the released data can be archived and accessed retrospectively. Prior approaches focused on various access heuristics and elastic expiration times to make the data unreachable to some extent. However, there are still many pending issues related to the proposed studies, such as securing ephemeral key storage and co-ownership data deletion. In this chapter, we attempt to tackle the problem of storing ephemeral keys during the estimated validity period. Hence, we devise a novel concept called key decay over time, which can achieve the ephemeral existence of the key. The decay idea entails the gradual, irreversible corruption of the key with time passing. In the current work, we combine the concept of gradual time elapsing and corruption into a single notion of the decay rate. Meanwhile, the irreversibility merit formed by randomness and various obfuscation strategies impedes retrospective attacks. Over time, the decay rate will give an estimated range for the key to be destroyed entirely. Finally, we implement and thoroughly assess a proof-of-concept regarding the key decay, including computational complexity and security analysis.

This chapter is based on the paper "Digital Forgetting Using Key Decay" by Darwish, M.A., & Zarras, A. in ACM/SIGAPP Symposium On Applied Computing 2023 (pp. 34-41) [56].

2.1. INTRODUCTION

Online data has been increasing rapidly, exponentially challenging our ability to manage and store it. IBM released a study that data growth witnessed a tremendous increment in 2020 to reach 40 trillion gigabytes (or 40 zettabytes), meaning every person on earth generates 1.7MB of data each second [57]. Meanwhile, according to Cisco, Internet consumers have reached 3.9 billion in 2018, and the number is expected to reach up to 5.6 billion in 2023 [58]. Nevertheless, the data owner may not need the online content to be accessible when its need ceases over time. On the other hand, the inadequate deletion practice of outsourced data jeopardizes plenty of security and privacy risks, including the possibility of data alteration or misuse by unauthorized parties and loss of confidentiality [43, 59].

The *General Data Protection Regulation* (GDPR) of the European Union enshrines the *Right to Erasure* or the *Right to be Forgotten* as it is widely known [60, 61]. This notion is gaining thrust and becoming extremely significant in protecting online privacy. Digital forgetting refers to the disappearance of data that has been uploaded to online storage platforms after it has fulfilled its purpose. In this sense, several underlying mechanisms have been developed to support digital data forgetting. For instance, cryptographic mechanisms have been proposed to encrypt the uploaded data in cipher form alongside the links. These links are the map for each data object to combine its key from ephemeral storage in order to be able to decrypt the online content within the validity period [62, 63]. On top of that, distributed architectures allow data owners to divide encryption/decryption keys in a distributed manner, using predetermined or flexible expiration periods, and bypassing single authority failures [44, 64–67]. These approaches utilize various services, such as the *Domain Name System* (DNS), the *Distributed Hash Table* (DHT), or website pages, to store ephemeral keys (i.e., keys with a short lifetime).

Unfortunately, the previous approaches depend on ephemeral storage for their decryption keys. This is problematic due to insufficient deterrence of the archival process. Even with DNS entries, the storage for ephemeral keys is still vulnerable to being cached and used retroactively [68]. As long as the ephemeral keys exist, anyone with access and sufficient resources, or the service provider, can cache them beforehand. The adversary will be interested in reaching out to the data after its validity has ceased. The crucial characteristic to be offered by any infrastructure is to impede and prevent keys collection on a large scale during the lifetime of the online content. This is challenging due to the availability of the keys to the public on the ephemeral storage.

In this work, we focus on excluding the necessity of the key storage and preventing the archival process from a retrospective adversary acting periodically by caching the whole data on the available distributed

infrastructures such as DNS. More specifically, we introduce the main idea of *key decay*. The decay concept is formed by three major ingredients: *time*, *corruption*, and *irreversibility*. Time implies a gradual sequence of events that follow one after another. Corruption entails changing from a state of soundness or perfection into a completely unusable and unsound state. Irreversibility is closely related to the notion of time in physics. This must be added separately due to the simplicity of reversing any temporal sequence in computer science without adding extra means to ensure such irreversibility.

As such, encryption is based on a key powered by random numbers to obtain a unique output called a seed. The creation seed of the key depends on a value that gradually changes from an old state, resulting in an entirely new one. Finally, the key irreversibility concept consists of *randomization* and *obfuscation*: different elements are combined in randomization to obtain miscellaneous values from online sources, while obfuscation complicates the origin key by crystallizing several techniques.

In essence, our approach reinforces the online data deletion by ensuring the transience of the key without counting on any ephemeral storage. The creation seed (i.e., randomly generated value) for the key will be obtained from online data, such as YouTube views, Twitter reposts, and Facebook likes. These online values will be represented as x_i coordinate in the Lagrange-basis polynomial [69]. Gradually, these online data values will modify, leading to utterly different values. This evolution of the values over time will coin the corruption concept. This corruption process for online values leads to a key forgetting in the end. For the key recovery, we use *Shamir Secret Sharing Schema* (SSSS) [70] to combine the correct parts during the reconstruction phase. The more corruption, the less key recovery as time goes on. Eventually, when the key decays, the irreversibility attribute prevents retrospective attacks from exposing the origin of the key. Moreover, the decay rate will provide insights into the expiration time of the key (i.e., days, weeks, years) according to the ephemerality level of the online data. We implement a system incorporating SSSS with the decay notion as a proof of concept. Introducing the key decay concept ultimately prevents the retrospective impact of an attacker or cloud provider that aims to store private keys and use them afterward.

In summary, we make the following main contributions:

- We propose using key decay for digital forgetting and prove how this will provide security against retrospective attacks.
- We design a scheme that guarantees the key ephemerality without relying on ephemeral storage.
- We evaluate our key decay approach by implementing a prototype.

Our results illustrate the decay rate in terms of evolution, stability, irreversibility, and computational complexity.

2.2. THREAT MODEL

Existing data deletion schemes [44, 64–67] frequently adopt the following procedure: (*i*) Data owners outsource their data to service providers after encrypting the content. In addition, the owners provide the corresponding links necessary for compiling the decryption keys at the recipients' end. (*ii*) During the validity (i.e., a dedicated lifetime of the data), within a specific period stated by the owners (i.e., fixed or flexible), the recipients with valid access can combine the key bits to decrypt the online content. (*iii*) After the validity, no one should be able to access the encrypted data object except those who already have private keys. In principle, those schemes generally leverage ephemeral storage such as cache entries, including DHT, website ciphers, and the DNS cache to support digital oblivion.

The main objective of our approach is to impede the retrospective adversary and fulfill online privacy. As such, we assume an adversary who is only curious about accessing the online content after its expiration. We also assume a cloud provider, or any other party, who periodically caches the key in the ephemeral storage. Snapshotting all the shared content can defeat the concept of digital forgetting. Owners should be conscious that the provider's storage and machines do not belong to them. Eventually, keys will be used to decrypt the content and expose privacy across the platform. In essence, data objects will be present even after the key is destructed. Once the dedicated expiration time is reached, acquiring the keys will help an adversary reveal the content again.

2.3. HIGH-LEVEL IDEA

This section discusses the research goals and the general view of our proposed scheme.

2.3.1. SECURITY GOALS

We must consider several security aspects to make a proposed solution as robust as possible. Therefore, we formalize our security goals in the following section.

SG1 – ACHIEVE RETROSPECTIVE PRIVACY:

An attacker must be incapable of accessing the data when the time is due.

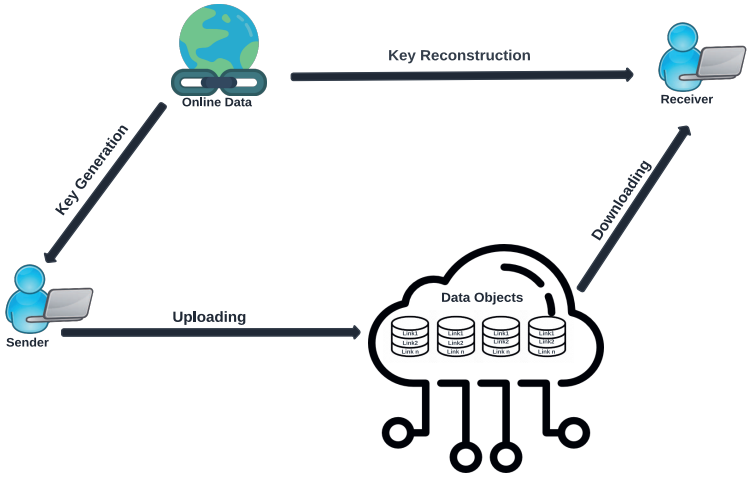


Figure 2.1: Proposed System Model.

SG2 – ENSURE THE EPHEMERALITY OF THE KEY:

Decryption keys should be destructed automatically by excluding the repository that stores them the whole time.

SG3 – DATA CONFIDENTIALITY:

Secure the data objects that will exist over the provider machine when the key decays after a certain period of time.

SG4 – ANTI-ARCHIVING:

Hinder the service provider to cache or archive the ephemeral keys on a large scale during the content's lifetime.

SG1 pertains to retrospective privacy by successfully preventing access to a data object after its expiration. *SG2* and *SG3* guarantee that the key will be self-destructed over time. *SG4* prevents attackers or cloud providers from taking a snapshot of a sheer number of data objects with their keys. Caching all the sources in a way to generate a key for each uploaded object is an impossible way to reach. A large-scale attack is out of the question for the adversary.

2.3.2. ABSTRACT ARCHITECTURE

Our scheme aims to enable digital forgetting by encrypting content and then generating and reconstructing ephemeral decryption keys without needing temporary storage for the keys. In more detail, the sender utilizes the key generated by our scheme to encrypt specified content

Table 2.1: Summary of notation.

Notation	Description
x, y	Coordinates of the polynomial
K	Threshold of the polynomial
N	Total number of the polynomial points
l	Lagrange-basis polynomial
$C(k, n)$	Key recovery combination
n	Total number of leading zeros

before uploading it. The scheme uses online values to create the key. The creation seed of the key counts on Lagrange basis polynomials to be utilized in both encryption and decryption phases [69]. After that, the creation seed will be utilized as an input for the AES algorithm to generate the private key [71]. Eventually, the recipient with valid access only will reconstruct the key similarly to obtain the original file. Figure 2.1 illustrates the high-level architecture of our proposed scheme.

2.4. SCHEME DESCRIPTION

This section delves into the details of the proposed system model. Our scheme-based key decay consists of four main steps: (*i*) *K*ey Generation Phase, (*ii*) *E*ncryption Phase, (*iii*) *R*econstruction Phase, and (*iv*) *D*ecay Phase. Table 2.1 provides a convenient summary of the notation used in this section.

2.4.1. KEY GENERATION PHASE

At this stage, data owners need a key to encrypt their content before uploading it. For each new key, our scheme performs the following:

Step 1. Our system depends on randomization to derive the key. The creation seed of the key will utilize various online sources (i.e., YouTube, Facebook, Twitter) to get certain types of values (i.e., views, likes, tweets). To this end, the scheme will produce a wide range of online values to be fitted within the Lagrange polynomial.

Step 2. After achieving the list of online values, they will be used to produce a random polynomial. The points are $P_i(x_i, y_i)$, where x_i represents the online values (i.e., views, likes, tweets), and y_i represents randomly generated values to fit the polynomial. In our case, Lagrange polynomial interpolation consists of two aspects: the **threshold** K, which means the minimum required number of points to produce the polynomial, and the **total number of points** N, which means the maximum number of points to produce the exact polynomial.

Step 3. From the online values mentioned above, the polynomial will take only K points (i.e., threshold) to be fitted. In principle, the Lagrange basis polynomial from K degree (i.e., minimum points) will be:

$$l_j(x) = \prod_{i \neq j, 0<i<k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \dots \frac{(x - x_k)}{(x_j - x_k)}. \tag{2.1}$$

By using this formula to generate the polynomial interpolation relying on the minimum points K, $F(x)$:

$$F(x) = \sum_{j=0}^{k} y_j l_j(x). \tag{2.2}$$

As a convention, the creation seed will be coefficient-free after getting the formula. The obtained polynomial represents this point's seed, $F(0)$.

Step 4. After the threshold (i.e., minimum points) and the creation seed are achieved, we use the same manner as in SSSS to generate the total number of shares (i.e., maximum points) to be represented within the exact polynomial interpolation. The reason behind generating more points is to increase the possibility of retrieving the polynomial of degree K. To do that, by using equation 2.2 and a new set of online types (i.e., views, likes, tweets), $P_n(x_1, x_2, ..., x_n)$, to be replaced in the formula to get $P_n(y_1, y_2, ..., y_n)$ as points/shares N belonging to the same polynomial. Consequently, the achieved polynomial will contain a total number of points, and any K out of N will be sufficient to produce the same polynomial.[1]

Step 5. Instead of using the actual scraped values from the online sources, a hashing function (i.e., SHA512) will be used to output a fixed-size hash. To achieve this, each scraped attribute (e.g., views, likes, repost counts) is first converted into a proper integer representation. These integers are then concatenated and fed into the hash function, ensuring that heterogeneous online values are normalized into consistent numerical inputs suitable for use as x_i coordinates in the polynomial.

Proof-of-Work (PoW) is an additional mechanism to increase the computational complexity against brute force attacks [72]. For both x_i and y_i coordinates, the hash of a leading zero will be calculated from the binary value. The following equation refers to the number of leading zeros needed to get the target hash, where the condition can be equivalently expressed using a regular-expression style constraint (e.g., $hash \in \{0^n.*\}$):

$$hash[:n] = SHA512(x_i, y_i). \tag{2.3}$$

[1] Both K and N values should be pre-selected by the user

Here, n specifies how many leading zeros are required at the start of the hash digest (e.g., if $n = 4$, a valid output must begin with "0000..."). This ensures that externally sourced values are deterministically converted into proper x_i and y_i inputs, while the PoW constraint enforces additional unpredictability and computational cost.

Step 6. On top of PoW, we also apply other obfuscation techniques (i.e., alternating sums, mod are explained in Section 2.5) to confuse the attacker's predictability and prevent retrospective brute-forcing.

Step 7. Eventually, the key will be derived after getting the seed from the Lagrange.

2.4.2. ENCRYPTION PHASE

The sender will outsource the online content to the service provider at this juncture.

Step 1. Once the key is derived (i.e., from the creation seed), the data owner uses it to compile an *Encrypted Data Object* (EDO), which will later upload to the cloud. We use AES to encrypt the data; AES is a symmetric block cipher scheme. It uses keys of 128, 192, and 256 bits to convert these individual original blocks. After encrypting these blocks, it combines them to generate the ciphertext. In our scheme, we use a 256-bit key of the AES algorithm to encrypt the online content, as it is more secure than smaller lengths.

Step 2. Another prerequisite is the checksum, which will be calculated to detect any errors and verify the integrity of the original file later. We use the SHA512 hash to generate the checksum from the uploaded file.

Step 3. After encrypting the data, we get the EDO stored in the cloud provider. The EDO structure includes the encrypted data, the used points to generate the polynomial (x_i, y_i), and the checksum value from the original file. Figure 2.2 depicts how an EDO looks like.

2.4.3. RECONSTRUCTION PHASE

During this phase, the recipient with valid access will reconstruct the key to decrypt the online content.

Step 1. The main task is to assemble the points for the Lagrange basis polynomial to reconstruct the seed. The data object (i.e., EDO) consists of two sections: the online values (i.e., link, type) to represent x_i and a list of values to represent y_i. For the actual polynomial construction, the minimum points (K) out of the total generated points (N) are sufficient for the correct reconstruction. The following formula considers the possible threshold combinations for the seed:

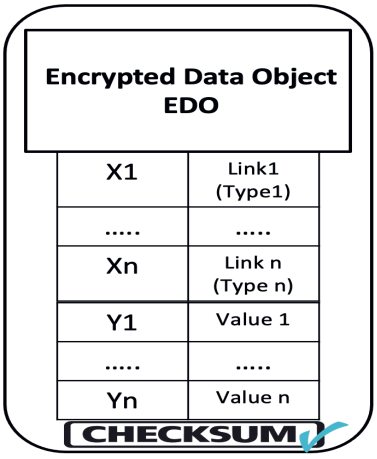


Figure 2.2: Data Object Structure.

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!(n-k)!}. \tag{2.4}$$

Step 2. After reconstructing the seed from the polynomial formula $F(0)$, the key will be reconstructed successfully. Then, the receiver can decrypt the content into its original form again.

Step 3. Over time, some values of the online data (i.e., views, likes, tweets) may differ from the old ones, as the central idea of the proposed scheme is to decay the key. The checksum validation will ensure the file's integrity from the selected threshold out of the total number. If the checksum is not matched, the scheme will provide a new threshold combination $C(n,k)$ set to reconstruct the key until the match is fulfilled.

2.4.4. DECAY PHASE

The decay phase is an essential concept of our proposed scheme. The validity period to recover the key falls between the threshold and the total number of points (shares). In contrast, the key recovery will no more applicable if the threshold is broken (i.e., $K-1$), leading to a key decay. The following inequalities show both validity and decay intervals.

$$K \leqslant Validity \leqslant N, \tag{2.5}$$
$$0 \leqslant Decay \leqslant K-1. \tag{2.6}$$

Over time, the online values will change, affecting the x_i coordinates used to construct the polynomial curve. This change will create new

(i.e., corrupted) points that are different from the old ones. Our scheme will capture these changes and present a different K combination from N to achieve the seed and eventually the key. In case of corruption has affected more K combination set than the limit (see inequality 2.6), this will cause the key to decay.

In a nutshell, the proposed methodology is inspired by SSSS. Shamir's one starts by having the key and creating the total number of shares and the threshold to divide it. However, our approach varies as online values will construct a Lagrange polynomial of degree K to obtain the seed. Then, generating more points as the total number of points. Correspondingly, PoW will be incorporated to increase the complexity of large-scale attacks. Moreover, the randomization notion is fulfilled by relying on different online values. Key traces will vanish over time as the values will keep corrupting till the threshold K is broken, leading to the key decay. Even though the data object will be present, the key will be no more in hand, which accomplishes the main concept of promoting digital data forgetting.

In this work, *key decay* refers to the decreasing ability to reconstruct the encryption key over time due to the progressive invalidation of the (x_i, y_i) points used in polynomial interpolation. The key material remains unchanged; instead, the recoverability decays, becoming computationally infeasible once fewer than K valid points are available.

2.5. EVALUATION

We have implemented a simulation prototype in Python to prove our proposed scheme's feasibility. Our prototype can dynamically provide values inbound of billions to simulate the online sources. Similarly, additional modules (Math, Random, Statistics, Itertools) simulate the Lagrange polynomial and the threshold combinations from the total number of shares. Cryptodome and Hashlib are used to encrypt and decrypt the content with the AES algorithm and PoW integration. The machine used for this evaluation is a MacBook (Processor: 2.3 GHz Dual-Core Intel Core i5, Memory: 8 GB).

Our experiments focused on five main aspects: randomization, decay rate, computational complexity, interpolation efficiency, and security analysis. Finally, we demonstrate the limitations of the proposed scheme.

2.5.1. RANDOMIZATION

As we mentioned (Section 2.4), the randomization notion is one of the crucial parts of our proposed scheme. In the real world, online sources provide various APIs (powered by pagination, different query order per request) to retrieve types (i.e., views, likes, tweets) [73–76]. To simulate

that in our proposed system, we relied on different random generators inbound between a million and a billion to get the same values expected from the actual scenario. We even defined a scraper for limited sources (i.e., YouTube) to acquire the exact response with all the types included (i.e., views, duration, tags).

2.5.2. DECAY RATE

The rate represents the total increment of the x_i value to decay over time, which in our context corresponds to *entropy*—the natural variability introduced by continuously changing online sources (e.g., view counts, likes, reposts) used in the creation seed, as described in Section 2.4. Since these online values evolve unpredictably, they act as random sources, ensuring that the coordinates (x_i, y_i) gradually diverge from their original state.

The practical impact of the decay rate on the key's lifetime depends on the relationship between the threshold K and the total number of points N. When $N-K$ is small (e.g., $K = N-1$), even a single corrupted point will render the key unrecoverable, making the decay rate less relevant. Conversely, when $N-K$ is large, the decay rate directly influences how quickly the system transitions from having at least K valid points to falling below that threshold.

From a practical standpoint, the choice of K and N directly governs the effective validity period of the key. A configuration with a small $N-K$ yields a **short-lived key**, suitable for ephemeral secrets that must expire quickly. In contrast, a larger $N-K$ yields a **long-lived key**, allowing more tolerance before the threshold is crossed. Furthermore, system designers can tune not only K and N, but also the type of external values sourced: fast-changing signals (e.g., trending counts or live sensor readings) yield keys valid for only minutes, while slower sources (e.g., daily statistics) extend validity to hours or days.

Furthermore, three folds of the decay were studied, (*i*) Evolution (i.e., the specific increment where the decay rate is due to forgetting the ephemeral keys), (*ii*) Stability (i.e., elapsed time taken for the key to transform from valid into decayed), and (*iii*) Irreversibility (i.e., the impossibility of reversing the operation to expose the origin of the key).

EVOLUTION.

We measured the decay rate with different types of evolution. By doing so, the incremental values will corrupt the key slowly till reaching complete decay. The rates studied vary between static rates (i.e., moderate and high $(+100, +10000)$ respectively per minute) and exponential rates (i.e., e^{x_i} per minute). Figure 2.3 displays how different function influences decay in various fashions, starting from exponential to a more direct static approach. The main reason for this experiment

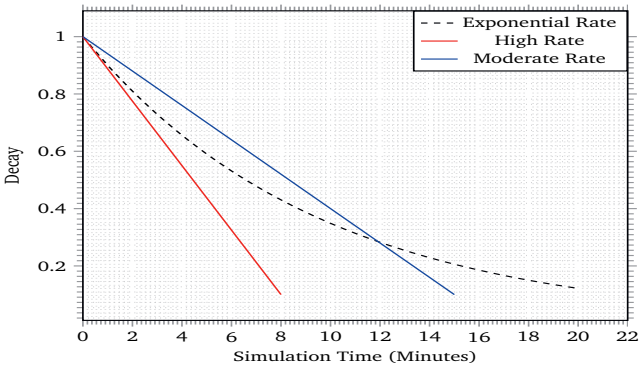


Figure 2.3: Decay Rates.

is to simulate the positive relationship between corruption level and ratio/fixed increment over time. The following equation explains the decay factor for a key, where λ refers to the corruption rate:

Table 2.2: Static Decay Rates.

Decay Rate	**Increment Per Minute**
Low	0
Moderate	100
High	10000

$$Decay = 1 - \lambda. \tag{2.7}$$

In our experiments, we decided to go with static rates as follows: (*i*) *Low Rate*, which means no change over time; (*ii*) *Moderate Rate*, which means an average increment over time; (*iii*) *High Rate*, which means a high increment over time. Table 2.2 shows the average increment over time (i.e., per minute in our case).

However, the static decay rates lead us to the reliable destruction of the key based on the parameters studied to initialize the rate, and forget the key over time. Furthermore, we focused on different combinations in order to measure the decay rate with the passage of time. Figure 2.4 shows the Lagrange polynomial before and after the decay rates (i.e., moderate and high rates) where the threshold K was surpassed, and the Lagrange polynomial has deviated from the original set of points. *In this example, only a subset of* x_i *values was altered to enable visual comparison between original and decayed polynomials. The pronounced ascents near the edges arise because a degree-14 Lagrange interpolant with evenly spaced nodes must pass*

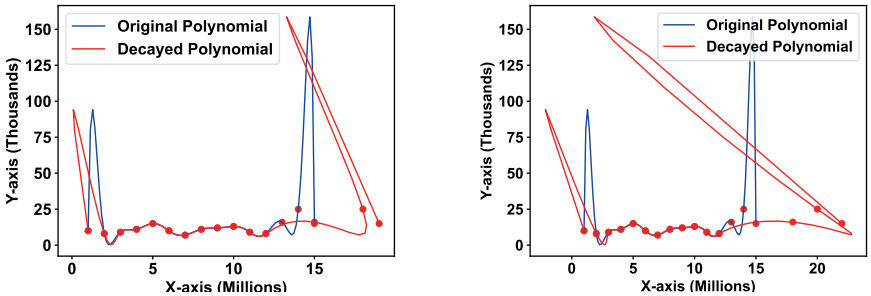


Figure 2.4: Moderate & High Decay Rates Using Means For 100 Samples of $(15, 50)$ Lagrange-Basis Polynomial.

through comparatively large endpoint y-values, forcing steep boundary growth before matching smaller interior points [77].

Table 2.3: Decay Rate Combinations.

No. Combinations	Random Decay Rate (min.)
$C_1(3, 5)$	4
$C_2(5, 10)$	6
$C_3(10, 20)$	11
$C_4(15, 30)$	16
$C_5(20, 40)$	22
$C_6(25, 50)$	26

STABILITY.

We addressed two sides of stability as follows:

1. WITHOUT USING MEANS (ORIGINAL):

This entails using only one type (i.e., YouTube view) to represent the x_i coordinate of the $P_i(x_i, y_i)$. This one value will increase immediately after a limited period leading us to complete decay in no time. Table 2.3 shows the different combinations $C(k, n)$, and the elapsed time taken to corrupt the threshold using either one of the decay rates. The main reason for the selected combinations $C(k, n)$ is to cover various samples that will not add stability to the scheme by relying on one value. Also, the threshold was selected according to equation 2.4 to reach the highest possibility (i.e., high combinations acquired) of recovering the key.

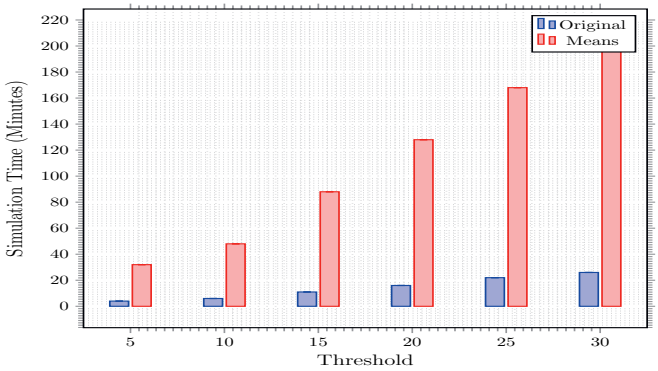


Figure 2.5: Stability Comparison With/Without means Usage.

2. USING MEANS:

From a different viewpoint, using means entails taking a sheer number m of values to represent the x_i coordinate of the $P_i(x_i, y_i)$. In our case, x_i is computed as the truncated arithmetic mean of m external values at fixed precision d decimal places: $x_i^*(t) = \text{trunc}_d\left(\frac{1}{m}\sum_{j=1}^{m} v_j(t)\right)$. This aggregation improves stability because a change in a single source value by Δ shifts the mean by only Δ/m. Thus, x_i^* changes only when the total cumulative change reaches 10^{-d}, making individual source fluctuations negligible for large m.

Example: With $m = 10^6$ and $d = 3$, x_i^* is truncated to three decimal places. For instance, $1.2349 \mapsto 1.234$ and $1.2341 \mapsto 1.234$, meaning small variations do not immediately change the stored value. As a result, a change in x_i^* requires a total cumulative shift of about 1000 across all m sources. At a moderate decay rate (+100 per minute across all m sources), this takes roughly 10 minutes in the worst case before x_i^* changes, significantly prolonging stability compared to using a single source. Table 2.4 shows the elapsed time taken to corrupt a single x_i value using different numbers of samples.

Table 2.4: Mean Samples.

No. Samples	Moderate Rate (min.)	High Rate (min.)
10	8	3
50	45	16
75	69	25
100	80	35
150	110	40

Let us assume we defined ($K = 15$ and $N = 50$), respectively. For a

moderate decay rate of corruption, each x_i coordinate takes the mean output of 100 online values to represent its coordinate. It took 80 minutes to corrupt one x_i value. According to our experiments, the threshold is 15. It is applicable to mutate up to 35 points as a maximum to recover the same seed again. The total time to corrupt the threshold and retrieve a different value is roughly 2800 minutes. For a high decay rate of corruption, it took only 35 minutes to corrupt one x_i value. As a result, a total of 1200 minutes is enough to surpass the threshold leading to a key decay. For more experiments, Figure 2.5 compares the stability time with/without using means in terms of different threshold samples (i.e., $[5, 30]$) over time, locating the same total number N.

IRREVERSIBILITY.

We highlighted two sides of irreversibility as follows:

1. USING ALTERNATING SUMS:

The use of means that always determine the cumulative average (between old and new values) will be expected at some point. Thus, using an alternating sum will confuse the attackers by switching between positive and negative signs of the mean outcome to get a less predictable polynomial; the following formula indicates that; where k represents the threshold and i represents the increment number of alternating the sign:

$$Alter(sum) = \sum_{i=0}^{k} (-1)^i \widehat{\mu}_i. \tag{2.8}$$

Figure 2.6 shows the alternating sums achieved from a polynomial after calculating the output with a negative sign. *For visual comparison, only a subset of x_i values was altered using alternating sums (red interpolated curve) against the original polynomial (blue curve). The unusual edge ascents occur because the high-degree interpolant must pass through large endpoint y-values, forcing steep boundary growth before aligning with smaller interior points [77].*

2. USING MODUOLO:

From a different angle, the attacker could determine the order of the polynomial by linking the points to get a predictable path. This will reduce the unknown possibilities of the shares since all points lie on a smooth curve. By using the following formula to fill the exposed values (i.e., $K-1$) from the threshold, respectively; where K represents the threshold, a represents the coefficients, and the seed represents $F(0)$:

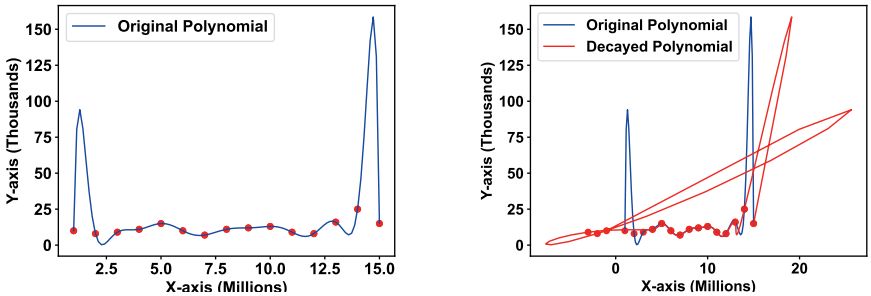


Figure 2.6: Decay Rate Using Alternating Sums for 100 Samples of $(15, 50)$ Lagrange-Basis Polynomial.

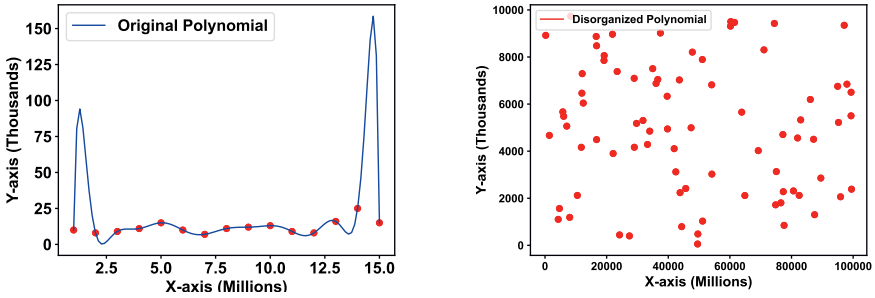


Figure 2.7: Difference Between $(x, F(X))$, and $(x, F(x)\ mod\ p)$.

$$F(x) = seed + a_1 x + a_2 x^2 + ... + a_{K-1} x^{K-1}. \tag{2.9}$$

After calculating the formulas (i.e., replacing and subtracting the exposed values), the attacker will obtain a fixed number range that will be used to guess instead of the infinite quantity of natural numbers. The attacker will narrow the prediction to expose the seed $F(0)$. To tackle this issue, finite field arithmetic will be used $p \in \mathbb{P} : p > N, seed$. The polynomial will be calculated as $(x, F(x)\ mod\ p)$ instead of $(x, F(x))$. Figure 2.7 represents the disorganized and disjointed polynomial using mod compared to the smooth one.

In light of the results, the low rate keeps the content for a very long period, as there is no change. The threshold and the total number of points can be deduced from the previous experiments to estimate the expiration times for the online data forgetting for both moderate and high rates. The relationship between decay and recovery rate is inverse: the less threshold with more points, the more applicability for

key recovery, contrary to the decay rate. Eventually, implementation with/without means alone is inadequate for the real-world scenario to fulfill the irreversibility. Thus, we propose the obfuscation strategies (i.e., alternating sums and mod) to construct disorganized and unpredictable Lagrange polynomials.

2.5.3. COMPUTATIONAL COMPLEXITY

The complexity of the proposed scheme means the amount of time and resources needed to achieve the target. PoW is an additional requirement to increase the computational complexity of the hash and make it time-consuming to obtain. Leading zeros are requested to find the optimal nonce (i.e., increment value) that matches the target hash with specific zeros [72]. Table 2.5 shows the different leading zeros implemented in our proposed framework to acquire only a single x_i value.

Table 2.5: Complexity Measurement.

Number of Iterations	Time Elapsed (sec.)	Leading Zeros
27	0.000103	$Hash[:1] = 0$
601	0.00119	$Hash[:2] = 00$
21674	0.0206	$Hash[:3] = 000$
22748	1.10501	$Hash[:4] = 0000$
148454	600	$Hash[:5] = 00000$
890724	2881	$Hash[:6] = 000000$

The study shows the total iterations and time elapsed to get only one hash for a specific x_i value. The number of leading zeros determines the difficulty. Acquiring a proper nonce to compute the hash is time-consuming; instead, the confirming part is relatively easy.

2.5.4. INTERPOLATION EFFICIENCY UNDER ENTROPY DECAY

Interpolation plays a crucial role in SSSS and cryptographic key reconstruction. Under entropy-driven decay, shares degrade over time, requiring a more efficient interpolation method. This section compares Standard Lagrange Interpolation and Barycentric Lagrange Interpolation with Entropy-Based Weighting and evaluates their performance in AES key reconstruction.

STANDARD LAGRANGE INTERPOLATION

Given $n+1$ points $(x_0, y_0), \ldots, (x_n, y_n)$, the standard Lagrange interpolating polynomial is:

$$P(x) = \sum_{i=0}^{n} y_i \ell_i(x),$$

2

where each Lagrange basis polynomial is:

$$\ell_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}.$$

To incorporate entropy decay, we introduce a soft weighting term that reduces the influence of high-entropy (i.e., unreliable) shares. The weighted interpolation becomes:

$$P(x) = \sum_{i=0}^{n} \tilde{w}_i \cdot y_i \cdot \ell_i(x),$$

where the normalized entropy-aware weight $\tilde{w}_i$ is defined as:

$$\tilde{w}_i = \frac{e^{-\lambda H_i}}{\sum_{j=0}^{n} e^{-\lambda H_j}}.$$

Here, H_i represents the entropy associated with the share at x_i, and λ is a tunable decay parameter that governs how aggressively entropy influences weighting. As entropy increases, a share's weight in the interpolation diminishes.

Key reconstruction remains possible as long as at least k shares remain sufficiently low in entropy. If more than $n - k$ shares are degraded beyond usability, the interpolation fails, and the key cannot be recovered.

BARYCENTRIC LAGRANGE INTERPOLATION

Barycentric Lagrange interpolation is a numerically stable and computationally efficient alternative to standard Lagrange interpolation [78, 79]. The interpolated value is given by:

$$P(x) = \frac{\sum_{i=0}^{n} \frac{w_i y_i}{x - x_i}}{\sum_{i=0}^{n} \frac{w_i}{x - x_i}},$$

where the classical barycentric weights are precomputed as:

$$w_i^{\text{bary}} = \frac{1}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}.$$

Barycentric interpolation offers several advantages:

Table 2.6: AES Key Reconstruction Time (ms) Across Standard and Barycentric Variants.

Shares	AES-128 Standard	AES-128 Barycentric	AES-192 Standard	AES-192 Barycentric	AES-256 Standard	AES-256 Barycentric
16	0.89	0.42	1.13	0.51	1.45	0.66
32	1.72	0.79	2.08	0.93	2.51	1.12
64	3.44	1.58	4.21	1.87	5.02	2.24
128	6.89	3.10	8.42	3.69	10.12	4.42
256	13.82	6.24	16.73	7.43	20.24	8.91
512	27.48	12.36	33.02	14.72	39.74	17.63

- **Computational efficiency**: Avoids redundant computation of basis polynomials.
- **Numerical stability**: More stable for high-degree polynomials compared to standard form.
- **Efficient share updates**: New shares can be incorporated without re-evaluating the full polynomial.

To incorporate entropy decay, we apply entropy-based soft weighting to the barycentric weights:

$$\tilde{w}_i = \frac{e^{-\lambda H_i}}{\sum_{j=0}^{n} e^{-\lambda H_j}}, \quad w_i = \tilde{w}_i \cdot w_i^{\text{bary}}.$$

Entropy H_i evolves over time and can be modeled using Shannon entropy:

$$H_i(t) = -\sum p_{ik}(t) \log p_{ik}(t),$$

where $p_{ik}(t)$ is the time-dependent probability distribution over possible values of share i. As H_i increases, the share becomes less trusted in interpolation.

Key reconstruction becomes infeasible when cumulative entropy degradation causes insufficient usable shares (i.e., fewer than k). The entropy-weighted Barycentric interpolation method thus offers a scalable, adaptive mechanism for handling share decay under entropy constraints.

PERFORMANCE EVALUATION: AES KEY RECONSTRUCTION

The AES seed generation relies on interpolation to reconstruct secret values. We evaluate performance by measuring the reconstruction time of AES keys using both Standard Lagrange Interpolation and Barycentric Lagrange Interpolation, comparing their efficiency under entropy-based decay conditions (see Table 2.6). Barycentric interpolation is consistently faster than standard Lagrange, with efficiency gains increasing as the number of shares grows.

2.5.5. SECURITY ANALYSIS

We analyze the proposed approach regarding different security goals and threats.

RETROSPECTIVE PRIVACY.

All physical data control is handed to the service provider when information is published on the Internet; data owners do not own the server storage. Also, the service provider or anyone can access and store all the keys within the ephemeral storage. However, our scheme provides a key decay notion without counting on ephemeral storage. Under those circumstances, the attacker will be incapable of performing a retrospective attack. The data will be secure, and retrospective privacy will be completely attained.

BRUTE FORCE ATTACK.

An attack on all possible x_i coordinates (online values) involves guessing the old value according to the data creation time and trying to brute force the remaining values retrospectively. PoW was integrated within the scheme to prevent such attacks on a large scale and provide more durability for seed generation. In our prototype, we investigated the elapsed time taken to exploit a single value x_i (see Table 2.5).

PREDICTABILITY.

An attack to predict the incremental values that lie on the smooth polynomial and expose the seed. To address this problem, the results section introduced alternating sums with mod usage and assessed them thoroughly. The outcome distinguishes the difference by using these concepts to make the polynomial less predictable and more disorganized (i.e., disjointed) to be used in the real-world scenario (see Section 2.5.2).

CURIOUS-BUT-NON-INTERFERING ATTACKER.

This attacker aims to snapshot the public data indiscriminately before its expiry for malicious use. The snapshot process will happen periodically

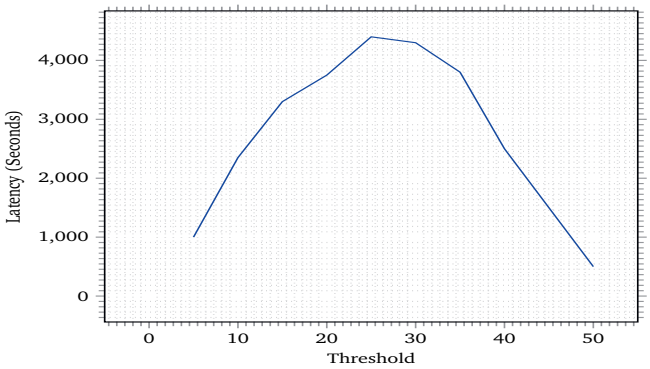


Figure 2.8: Latency For Different Thresholds.

(i.e., daily, weekly) to extract sensitive information. Our scheme relies on two central notions: randomization and the exclusion of ephemeral storage. Thus, this will hinder the curious attackers from caching and archiving the data periodically to keep them for future use.

2.5.6. LIMITATIONS

As with any other work, ours has its limitations, which are demonstrated below.

LATENCY.

As we mentioned in the scheme description, the checksum will be validated during the reconstruction phase to get the correct polynomial seed $F(0)$. However, according to the combination calculation $\binom{N}{K}$, choosing a rate for the threshold close to half of the total number will give many possibilities for key recovery (see equation 2.4). This will add latency to the system to match the correct checksum by performing many attempts. Figure 2.8 shows the total latency added to the scheme by performing different samples of thresholds between $[5, 50]$. From a practical perspective, this latency can be mitigated by selecting K significantly smaller or larger than $N/2$ when near real-time reconstruction is required, or by parallelizing the checksum validation process. Designers of the system should balance this trade-off between latency and the combinatorial recovery complexity based on the intended security and performance requirements. While these strategies highlight possible directions, developing more optimized approaches to reduce recovery time remains an open problem for future work.

ATTACKER WITH HIGH COMPUTATIONAL POWER.

The most recent models of GPUs, which have capabilities for integer computations and specialized hardware like FPGAs, can be used by well-funded attackers to increase performance. According to certain studies, commercially available hardware can speed up public-key cryptosystems like the RSA scheme by about four times, which can then be applied to these challenges [80]. This will affect the power of the PoW technique by reducing the computational complexity introduced by our scheme.

2.6. RELATED WORK

This section briefly highlights the various current state-of-the-art studies that look into digital forgetting and provide an online data revocation between the data owners and the service providers. Thus, researchers have presented diverse solutions to compensate for the existing vulnerabilities and provide secure approaches to tackle this issue.

In this sense, the most common solution to cope with this vulnerability is to define the data's expiration time. Several techniques are proposed to help deal with this security issue, including flexible and predefined expiration periods (e.g., Ephemerizer [64], Vanish [65], Neuralyzer [66], EphPub [44], and Timed Revocation [67]. Another study [45] carried out the scheme using time-lock cryptographic puzzles to support digital forgetting. The essence of the proposed study is to limit the deletion or collection of the data during the lifetime of the content by adding complex overhead to the large-scale attack. The time-lock puzzle will provide proof of work to anyone that wants to access the data. Another approach [81] integrates an automated protocol for handling data revocation by examining contractual agreements between cloud providers and data owners. A protocol for this specification of revocation conditions was implemented using smart contracts on a local Ethereum blockchain. Yang et al. [82] have used a novel authentication data structure, namely, the number-rank-based Merkle hash tree (NR-MHT). The prime core of this research is to provide a provable data deletion scheme based on efficient data integrity auditing and dynamic data insertion. Xiong et al. [83] proposed a novel key derivation encryption algorithm that will be used to apply a secure deletion in the Internet of Things devices. The proposal is based on flash memory's hierarchical structure by partially merging the block's erasure and deleting the key. The deletion will be from both the cipher form and components that belong to the key when the data validity is ceased. On the other hand, Ginart et al. [84] proposed deletion efficiency for large-scale learning systems and possibly deletion-efficient unsupervised clustering algorithms. They identified potential algorithmic principles that may facilitate deletion efficiency for different learning

systems and paradigms. A lethe mechanism was introduced by Minaei et al. [85] to provide post-deletion privacy by withdrawing intermittently Twitter posts. The attacker will be confused between the temporary and permanent contents after a long time of the content deletion. The core of this mechanism is to provide a data owner's deniability against their online posts. This technique will prevent adversaries from archiving posts from the Twitter platform. Moreover, the Forgits data structure strengthens the online deletion by gradually dropping the lowest bits or pixels (least significant bits) from old to the most significant bits from new data [86]. This structure provides infinite retrievals by forgetting the older stored data. However, these approaches provide a digital forgetting-based third authority involvement to forget the desired data; they cannot eliminate the central need for third-party storage or actions. To the best of our knowledge, a key decay notion is a novel approach that has yet to be studied. In this work, we showed that it might be used to prevent adversaries from performing retrospective attacks on tailor-made data to be forgotten without the need for ephemeral key storage.

2.7. CONCLUSION

In this study, we proposed a novel scheme of *key decay* in the realm of digital forgetting. Because of the ubiquitousness and popularity of digital data, solutions that enable transientness and ephemerality are in high demand. Retrospective privacy is extremely critical since adversaries are keen to exploit the data after its expiration. Our approach introduced a decay vision that implies a change from soundness to complete corruption without counting on the key's storage throughout the period. Furthermore, using different resources to construct the key besides integrating the obfuscation strategies will impede the attackers from storing or taking a snapshot of the keys during the content's validity. Finally, we implement and thoroughly assess a prototype with promising results regarding decay rate, complexity, and irreversibility.

3

AUDIENCE-BASED EXPIRATION

The virtue of data forgetting has become a substantial demand in the digital era. Once online content has served its purpose, the concept of forgetting arises to ensure that data remains private between data owners and service providers. Despite significant advancements in supporting data forgetting through approaches like access heuristics, elastic expiration times, and manual revocation, the existing research falls short in addressing the demand for a multi-level forgetting structure that can cater to diverse audience-based expiration requirements while considering additional criteria. To the best of our knowledge, no prior works have investigated this gap, emphasizing the need for a comprehensive solution that can effectively accommodate the varying expiration needs of different audience groups. In this chapter, we introduce a novel disjunctive multi-level forgetting scheme designed to meet the aforementioned demand for data forgetting. Our scheme introduces unique expiration periods for the encrypted data the service provider stores, called levels. Users are grouped into different levels based on priorities assigned by the data owners. Each level corresponds to a specific expiration threshold, enabling designated user groups to access the content within its validity period before it is forgotten. This approach enables selective data forgetting for one group while enabling concurrent access and retention for other user groups until the stipulated expiration period elapses. To achieve this, we have devised a cutting-edge system that integrates a hierarchical and dynamic scheme utilizing a key decay for managing expiration periods. Moreover, we introduce an innovative approach that harnesses smart contracts on a local Ethereum blockchain to enforce regulations and streamline

This chapter is based on the paper "Disjunctive Multi-Level Digital Forgetting Scheme" by Darwish, M.A., & Smaragdakis, G. in ACM/SIGAPP Symposium On Applied Computing 2024 (pp. 112-121) [87].

the secure and efficient expiration and deletion of data. Finally, we thoroughly evaluate our proposed scheme, focusing on decay sensitivity, computational complexity, and rigorous security analysis.

3.1. INTRODUCTION

In our information-driven world, data storage and access have become incredibly efficient and affordable [88]. This era of "big data" has seen a remarkable surge in global data generation. According to the Internet Data Center (IDC) [2] report, global digital data jumped from 4.4 zettabytes in 2013 to an astounding 40 trillion gigabytes by the end of 2020. This exponential growth is projected to reach an estimated 175 zettabytes by 2025. While this abundance of data offers numerous advantages, it also poses real challenges. However, such data storage and utilization raise concerns regarding privacy and data protection [43, 59, 89]. As a response to these concerns, GDPR was enacted by the European Union, emphasizing the *Right to Erasure*, also known as the *Right to be Forgotten* [60, 61]. This concept aims to give individuals control over their data, allowing them to request the removal or deletion of their data from various platforms and databases [30]. To facilitate the process of digital data forgetting, several techniques have been proposed, including Vanish [65], EphPub [44], and Neuralyzer [66]. These approaches primarily focus on uploading online data to service providers in a manner that obscures its content, along with providing instructions on how to retrieve the private keys (i.e., decryption keys) during static or flexible expiration periods. They rely on ephemeral storage solutions like DNS [90] and DHT [91] to store and forget the private keys securely.

Unfortunately, previous studies [92–95] have predominantly focused on a single level of forgetting, where decryption keys are simultaneously accessible to all users during the designated validity period. Moreover, in these approaches, the decryption keys are entirely forgotten (i.e., destroyed) for all users once the validity period ends. In contrast, TAFC [96] enforces time-sensitive access control without actual key destruction, but it does not support audience-based expiration, as all users with matching attributes retain access until the predefined time elapses. However, in order to cater to the diverse requirements of different user groups, it is essential to implement privacy controls that allow for customized exposure management of ephemeral data. This means that decisions regarding data exposure should not uniformly impact all readers but should instead consider individual users or groups of users and their specific needs [43]. The need for more control in forgetting the private keys within a single-level scheme results in inflexibility and imposes challenges when disseminating content to multiple levels. By employing the same expiration periods for all

recipients, the ability to effectively manage the level of forgetting for stored data becomes limited [28].

Given these limitations, our research endeavors to overcome existing approaches' shortcomings by introducing an innovative ***Disjunctive Multi-Level Forgetting Scheme***. Our primary focus is presenting a multi-level structure that caters to diverse groups. Our novel scheme is constructed upon the foundation of the *key decay* framework [56], which ephemeral key embodies the concept of gradual deterioration from a state of integrity to complete corruption. In our approach, the multi-level scheme leverages a shared key, precisely the decryption key, associated with the same content but with distinct expiration times determined by owners. By allowing variations in the validity period across different groups, we aim to enhance flexibility and control over online content. Importantly, our proposed scheme achieves its objectives without needing central ephemeral storage for the keys or expiration periods. Our approach involves encrypting and uploading online content using the key decay concept. This encrypted content is then stored on the service provider's platform, where recipients with valid access privileges can download it. Our scheme categorizes recipients into distinct levels, each corresponding to a specific user group within the system. The levels in our scheme serve as discrete categories that include all users and are defined by specific rules, expiration periods, and validity periods to reconstruct the ephemeral keys. As a result, each user is assigned to a particular group based on the decision of the data owner. The data owner maintains the authority to determine the expiration periods by directly setting them or delegating this responsibility to the providers. Access to and downloading the content are contingent upon meeting the specified criteria for each group. This variation in expiration periods of the ephemeral key enables the same content to remain visible in one group while being forgotten simultaneously in another. As the decay progresses, all groups will forget the key, rendering the content useless.

In summary, our research makes the following contributions:

- We introduce the concept of multi-level key decay as a novel approach for achieving digital forgetting, showcasing its ability to provide flexibility and dynamism in data management.
- We develop a comprehensive scheme that ensures the key's partial and complete ephemerality, addressing the specific needs of different user groups.
- We evaluate the effectiveness of our disjunctive multi-level approach by implementing a prototype and conducting analyses on decay sensitivity, computational complexity, and security aspects.

3.2. CONCEPT

This section introduces terminology and design goals and provides a high-level view of the proposed scheme (see Figure 3.1).

3.2.1. TERMINOLOGY

We define the fundamental notions used in this study, which include:

Disjunctive Multi-Level Scheme. A hierarchical organization of audience (i.e., user groups), each with unique data expiration and access control. Users belong to a single group, unlike the conjunctive scheme, where they can be in multiple groups.

Sender. The entity responsible for encrypting and transmitting data, setting expiration periods, and generating the initial encryption key.

Receiver. The intended recipient with access to encrypted data is responsible for decryption and access based on expiration and group settings.

Encrypted Data Object (EDO). Data protected by the scheme, including encrypted data, key generation points (i.e., random sources), checksum, and predefined group rules.

Platform Provider. The service that hosts encrypted data provides infrastructure and tools for data management (e.g., Google Drive, Dropbox).

3.2.2. SECURITY GOALS

Numerous security factors must be considered to make a suggested solution reliable and feasible. Consequently, we formalize our security objectives in the following:

GOAL 1 – GUARANTEE A MULTI-LEVEL FORGETTING STRUCTURE.

The scheme should support the flexible configuration of expiration periods for different groups based on the preferences of the data owners.

GOAL 2 – ACHIEVE RETROSPECTIVE PRIVACY.

The scheme should prevent unauthorized access to the data only after the expiration of the decryption key, ensuring that expired content remains private[1].

[1] While this assumption may seem idealistic, it serves to demonstrate the concept of retrospective privacy. We recognize that real-world attackers may not always follow such protocols (e.g., not persistently storing unencrypted copies of the data) [43].

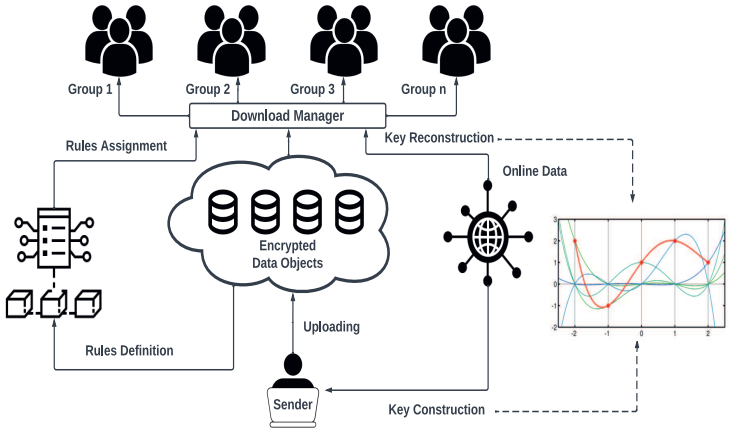


Figure 3.1: Proposed System Model.

GOAL 3 – ENSURE THE EPHEMERALITY OF THE KEY.

The scheme should ensure that decryption keys are wholly forgotten at a specific point. Achieving *Goal 1* results in varying key expiration times across different levels, enhancing system security and flexibility. *Goal 2*, retrospective privacy prevents attackers from retroactively exploiting data, maintaining privacy even if they gain access to expired data. Finally, *Goal 3* ensures secure key destruction across all groups, minimizing the risk of unauthorized access or decryption of data after key decay, ensuring long-term security.

3.2.3. OVERALL ARCHITECTURE

Our scheme provides multi-level forgetting periods built over a key decay scheme [56]. The data owner encrypts online content using a key generated from the decay scheme. Key generation involves fitting online data from random sources onto a Lagrange-basis polynomial [69]. After uploading the content, the download manager divides users into groups, each with its threshold for key reconstruction. This distribution process is facilitated through smart contracts, determining the key's lifespan. Each group can reconstruct the key during its designated period, decrypting the content. Our system allows data owners to dynamically set expiration preferences and easily assign or reassign receivers based on these preferences to different groups. As the decay period approaches, some groups experience key decay while others can still access it. Eventually, the key decays for all groups after a final period, achieving digital forgetting. This approach offers tailored audience-dependent expiration periods, enhancing data forgetting control.

Table 3.1: Summary of Notation.

Notion	Description
x, y	X, Y coordinates used in the Lagrange-basis polynomial
K	Threshold of the polynomial
N	Total number of the polynomial points
l	Lagrange-basis polynomial
$C(N, K)$	Key recovery combination
V	Validity period
D	Decay period
G	Group of level
L	Single level
μ	Stability mean
p	Prime modulus

3.3. SCHEME DESCRIPTION

This section thoroughly examines the proposed system model, covering four key aspects: (*i*) Key decay scheme background, (*ii*) Multi-level forgetting structure, (*iii*) Smart contract integration, and (*iv*) Expiration and group factors. Table 3.1 summarizes the notation used in this study.

3.3.1. BACKGROUND ON KEY DECAY SCHEME

This section introduces the core framework of our proposed system, the key decay scheme. This scheme utilizes *Single-Level Forgetting*, where the level represents a distinct group of users without subdivisions or overlapping. Users follow the same forgetting criterion for uniform decay. The implementation of the key decay scheme encompasses the following phases:

KEY GENERATION PHASE

To guarantee the encryption of the content prior to uploading, the decay mechanism relies on random sources to generate a seed for key creation. This creation seed utilizes diverse web sources, including platforms like *Facebook, Twitter, and YouTube*, to obtain specific values such as *views, likes,* and *tweets*. To accomplish this, the system employs a technique that generates a diverse set of online values (i.e., a set of points) capable of fitting into a Lagrange polynomial. The key generation process involves constructing a Lagrange basis polynomial, where each polynomial corresponds to a particular threshold K and the total number of points N that determine the decay rate. Once the ephemeral key is derived, it is employed in the AES 512-bit algorithm to facilitate the encryption phase [71]. This process involves

compiling and securely storing the resulting EDO in the cloud by the service provider. The system incorporates several techniques to ensure irreversible decay, including a consensus mechanism that mitigates predictability and increases computational complexity to deter brute force attacks (Proof of Work [72]). In addition, the system utilizes alternating sum and modulo operations to obfuscate the plain values obtained (random source points). The alternating sum operation denoted as $\sum_{i=0}^{k}(-1)^{i}\widehat{\mu_i}$, incorporates a series of switched sign values that contribute to the obfuscation process along with modulo operations represented as $F(x)\ mod\ p$, where $p \in \mathbb{P}$ is greater than N to produce disjointed polynomial [56].

KEY RECONSTRUCTION PHASE

To decrypt the online content for the intended recipients, the Lagrange polynomial is used to reconstruct the key out of EDO structure. The same sources used during the key generation phase are utilized to interpolate the identical polynomial by meeting the threshold K (i.e., minimum points) requirement. Through this scheme, the key is reconstructed, enabling the execution of the content in its original form.

KEY DECAY PHASE

To enable and streamline the process of digital forgetting, the key undergoes gradual modifications over time until it reaches a point of irrecoverability. As time progresses, the online values will change, consequently impacting the coordinates of the Lagrange curve used for key generation. These changes will introduce new corrupted points (i.e., N polynomial points) that differ from the previous ones. The decay scheme detects these alterations and generates a distinct combination of K out of N a ***fixed*** maximum number of points to obtain the seed. In the event that a more significant number of K combination sets are affected by the corruption, surpassing the predefined threshold, the key will decay. This decay will result in all recipients losing the ability to retrieve the key. As a result, the data will become inaccessible, meaning the encrypted data will still exist but will be rendered useless without a retrievable decryption key.

3.3.2. MULTI-LEVEL FORGETTING STRUCTURE

The proposed scheme includes *Disjunctive Multi-Level Forgetting*, wherein hierarchical levels are utilized to combine users into distinct groups. Unlike uniform decay, this scheme incorporates non-uniform decay, allowing for different rates of decay or expiration for various data or information within each group. The scheme expands the scope of the decay mechanism by serving two primary objectives: (*i*) **Partial Decay**,

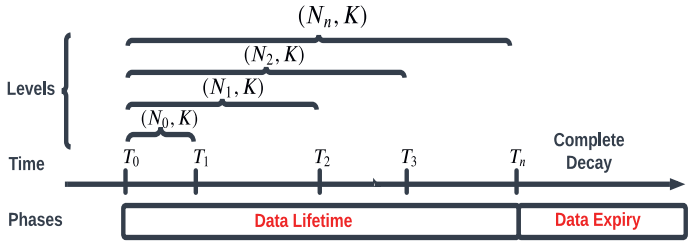


Figure 3.2: An Overview of The Multi-Level Forgetting Scheme.

enabling selective forgetting for each group, and (*ii*) **Complete Decay**, representing irreversible decay of the data for all groups. The overall framework of this multi-level scheme throughout the content's lifespan is depicted in Figure 3.2. The scheme comprises two main phases, outlined as follows:

UPLOADING PHASE

Our system incorporates a key decay scheme to generate the encryption key, also known as the seed. The key generation process relies on a Lagrange-basis polynomial, determined by two key parameters: K, the minimum number of points, and N, the maximum number of points. To construct the polynomial, we require a set of points, denoted as $P_i(x_i, y_i)$. Specifically, we are given a set of required points K: $(x_1, y_1), (x_2, y_2), ..., (x_k, y_k)$, where no two points have the same x_i within the set. The Lagrange-basis polynomial with a specified threshold is computed using a product formula:

$$l_j(x) = \prod_{i \neq j, 0<i<k} \frac{x - x_i}{x_j - x_i} = \frac{(x - x_0)}{(x_j - x_0)} \dots \frac{(x - x_k)}{(x_j - x_k)}. \tag{3.1}$$

The resulting polynomial is then used to interpolate the corresponding y values.

$$F(x) = \sum_{j=0}^{k} y_j l_j(x). \tag{3.2}$$

The value of $F(0)$ obtained from this interpolation is utilized to generate the required private key. The scheme will generate different levels of N to be fitted within the achieved polynomial, meaning that the total number of points will acquire several polynomial subsets. Using the Formula 3.2 above, $F(x)$ and a new set of online links (i.e., values), $P_n(x_1, x_2, ..., x_n)$, generate more points N belonging to

the exact Lagrange polynomial curve. Consequently, the achieved polynomial will contain a ***dynamic*** total number of points; any K out of N will be sufficient to get the exact polynomial. The main reason for this circumstance is that different N will be distributed and divided between recipients later in the decryption phase. Each sample will result in a unique key recovery period, leading to a different decay rate and a distinct expiration time for that particular combination. Each level contains one group (G) $Levels = \{L_0(G_0), L_1(G_1), \ldots, L_n(G_n)\}$. Once the key is derived, the online content will be encrypted using AES 512-bit before being uploaded to the provider.

DOWNLOADING PHASE

During this phase, authorized recipients are able to decrypt the online content. This process begins by reconstructing the key using the used points during content uploading. The download manager organizes users into groups based on their expiration periods, which are determined by user preferences. The combinations of key points, represented by the levels $L = C(N, K)$, are then distributed among the corresponding user groups. Each group must fulfill the requirement of having at least K points out of the assigned N_i (i.e., the subset of related polynomial points). Consequently, if one group has more points than another group, it increases the probability of key recovery, as indicated by the binomial coefficient of $\binom{N_i}{K}$ (permutations when $N = K$). To illustrate the range of key recovery possibilities across different levels, random combinations are generated. This process helps showcase the variations in the likelihood of successfully reconstructing the key within each level:

$$Possibilities = \begin{cases} C_0(N_0, K) \\ C_1(N_1, K) \\ \vdots \\ C_n(N_n, K) \end{cases}, N_i \geq K \geq 0. \tag{3.3}$$

This indicates that the multi-level structure offers the flexibility to define varying expiration thresholds for each group based on specific preferences and the number of possible key recovery combinations. The concept of ephemeral key bits revolves around the decay of particular values within the system after a certain period. This decay process relies on changing random sources to generate these values. Each group in the system has its own unique decay and validity interval, which distinguishes it from other groups. The inequality shown below illustrates the variation in these intervals:

$$\begin{cases} K \leqslant V_0 \leqslant N_0, & \text{if } 0 \leqslant D_0 \leqslant K-1, \\ K \leqslant V_1 \leqslant N_1, & \text{if } 0 \leqslant D_1 \leqslant K-1, \\ & \vdots \\ K \leqslant V_n \leqslant N_n, & \text{if } 0 \leqslant D_n \leqslant K-1. \end{cases} \tag{3.4}$$

Each level corresponds to a distinct combination of validity (V) and decay rate (D).Online content can be decrypted as long as the threshold for a particular level is not exceeded. However, decay occurs once the threshold is surpassed, rendering the online content invisible. This decay is referred to as *partial decay* for that specific level. Finally, when decay affects all sharing schemes (i.e., all subsets of the Lagrange polynomial $C_n(N_n, K)$), it results in *complete decay* for all recipients. Therefore, the key becomes irretrievable from the original Lagrange-basis polynomial. The central idea here is that the multi-level forgetting scheme offers flexible and dynamic control over distinct levels. Each group is independently isolated through the download manager to determine its specific validity and decay rate, resulting in unique expiration periods.

3.3.3. SMART CONTRACTS

Using Ethereum-based smart contracts, we streamline data expiration management, automate processes, enforce rules, and set expiration times for groups [97, 98]. This enhances sensitive data protection, reduces complexity and costs, and eliminates the need to store key traces or data fragments. Key management is off-chain, with only group-specific rules stored in Ethereum contracts. The formula below represents parameters for polynomial total points at each level ($L(G)$) with the same threshold:

$$Parameters = \begin{cases} P_0 = (L_0(G_0), K), \\ P_1 = (L_1(G_1), K), \\ \vdots \\ P_n = (L_n(G_n), K). \end{cases} \tag{3.5}$$

Furthermore, the smart contracts will enforce the shared parameters (i.e., the total number of points and user classification) for each group level to ensure the allowed validity is achieved. Each level can form a combination upon expiration by retaining the respective parameters, plus the same threshold shared between all the levels. The validity for each level (i.e., group of users) is described below:

Algorithm 1: Smart contract for rule-based key assignment and digital forgetting

Require: G (Groups per level)
Ensure : V (Validity periods for each level)

```
function
    Rule-Based Key Assignment()
Input: Divide users into G groups based on certain criteria by data
  owners;
for each level i do
    Define groups G_ij within level i; for each group j in level i do
        Assign point assignments N_xij for group G_ij; Define validity period
          V_ij for group G_ij;
    Assign a common threshold K_i for all groups in level i;
for each level i do
    for each group j in level i do
        Provide visibility to the content based on assigned combinations K_i
          and N_xij;
for each level i do
    for each group j in level i do
        if current decay > V_ij expiration time then
            PartialForgetContent(N_xij, K_i);
        else
            (N_xij, K_i, ) is still valid;
CompleteForgetContent();
```

$$V = C_i(N_i, K), where \binom{N_i}{K}, \tag{3.6}$$

$$C_i(N_i, K) = \frac{N_i!}{K!(N_i - K)!}. \tag{3.7}$$

In summary, our proposed approach involves a customized group rule assignment derived from the principles of EDO. This assignment includes the total number of points, and the level of access to a portion of these points is contingent upon user preferences. While a unified threshold is common across all groups, the potential for key reconstruction is based on the specific combination provided.

Algorithm 1 provides a detailed overview of parameter assignment (i denotes levels, j represents groups within a level, and x signifies point assignment variations within those groups). It begins by categorizing users into G groups based on specific criteria (data owners). For each group i, it defines levels L_{ij} and sets threshold values K_i, validity assignments N_{xij}, and validity periods V_{ij} for each level. The algorithm grants content visibility based on the assigned combinations K_i and N_{xij} for each group and level. When these combinations are determined, points are generated from an EDO and securely shared according to predefined rules, ensuring authorized user access. The contract then checks if the current decay has exceeded the expiration time V_{ij} for each group and level. Upon expiration, it triggers the **PartialForgetContent** function, which removes specific key combinations according to thresholds K_i and validity assignments N_{xij}. Afterward, the **CompleteForgetContent** function initiates comprehensive forgetting, deleting all associated combinations (ephemeral keys), offering a systematic approach for key assignment, validity management, and rule-based digital forgetting in a multi-level data framework.

3.3.4. EXPIRATION AND GROUP FACTORS

The proposed multi-level scheme integrates expiration factors and group classification for comprehensive key expiration [99, 100]. These factors and criteria regulate key validity and decay, categorized into levels assigned to user groups for customized expiration periods, enhancing key management and content control.

EXPIRATION FACTORS

Essential factors in determining expiration thresholds include:

Data Sensitivity. Influences the expiration period, shorter for highly sensitive data.

Security and Usability Balance. Ensures security and convenient access.
Data Nature. Varies based on data type, requiring tailored expiration.
Regulatory Compliance. Aligns with industry-specific regulations. In addition to the factors discussed earlier, our scheme incorporates the concept of key decay rate, known as "Evolution." This aspect is crucial for understanding how keys deteriorate over time, influencing our key management and content control strategies. Owners can customize evolution to align with group expiration periods stored and managed within smart contracts. We aimed to grasp how incremental values progressively corrupt keys until they decay completely. Our study encompasses three distinct decay rates(λ): (*i*) **Low Rate:** Implies minimal change over time, resulting in keys deteriorating very slowly, (*ii*) **Moderate Rate:** Involves an average increment over time, leading to a gradual decay of keys, and (*iii*) **High Rate:** Features a significant increment rate, causing rapid key decay. These decay rates enable effective key management, gradually obscuring and estimating approximate expiring keys based on chosen parameters and influencing factors.

GROUP CLASSIFICATION

Users are categorized into groups based on criteria like membership or access rights, allowing allocation of expiration thresholds. Two approaches are:
Owner-Defined Expiration: Owners set unique expiration periods for each group.
Membership Level-Based Expiration: Duration based on membership level or privileges.
The scheme offers fine-grained control over content expiration, enabling explicit rules or data-driven insights.

3.4. SYSTEM DESIGN AND ARCHITECTURE

With a focus on the multi-level forgetting scheme, the suggested system design, shown in Figure 3.3, offers a thorough overview of our framework. The system design begins with the data owners encrypting their data before uploading it to the provider environment. This encryption process ensures the confidentiality and integrity of the content. To enable the ephemerality of the encryption keys, the system incorporates a decay scheme based on Lagrange polynomials (Formulas 3.1 and 3.2). This scheme generates the keys randomly and determines various decay rates (i.e., low, moderate, and high) defined in smart contracts, allowing for controlled and gradual partial key corruption until complete deterioration. Upon EDO upload to the

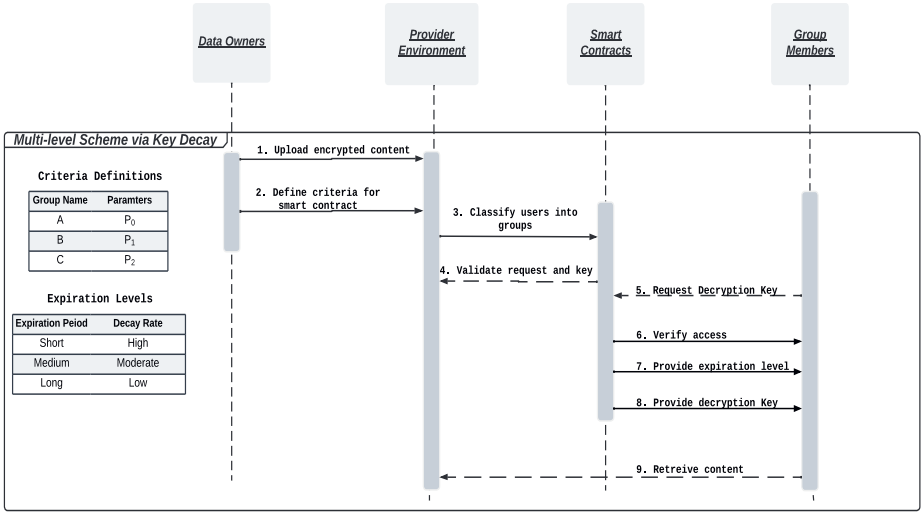


Figure 3.3: System Design and Architecture: Multi-Level Forgetting Scheme, Smart Contract Integration, and Expiration Control.

provider by compiling all points needed for reconstruction, owners can establish criteria for categorizing users into distinct levels and degrees of forgetting (see Section 3.3.4). This classification process enables personalized and finely-grained control over the assigned expiration levels for each group. The multi-level forgetting structure provides varying expiration thresholds tailored to user preferences or administrator decisions, ensuring a customized approach to key expiration. The group classification (G_i, $1 \leq i \leq G$) and corresponding expiration periods (E_i) for the G groups can be represented as:

$$E_i = \begin{cases} E_i & \textit{if Owner-assigned,} \\ f(m_i) & \textit{if Provider-assigned.} \end{cases} \tag{3.8}$$

These formulas capture the group classification and expiration periods, respectively. Each group is allocated a unique classification and expiration period, enabling personalized control over expiration levels based on user preferences. In cases where the administrator determines the classification (i.e., owners assign this role), the membership level m_i defines the rule or criterion utilized to ascertain the expiration period. Subsequently, the group classification information is transmitted to the smart contract, which acts as a decentralized authority responsible for managing and enforcing the defined criteria in EDO. Upon receiving the group classification information, the smart contract commences the distribution of expiration levels to the group members, as denoted by

Formulas 3.5 and 3.7. Each user or group within the system is assigned a unique identifier or profile. The smart contract initiates a confirmation request to the cloud provider to ensure accurate implementation and enforcement of the expiration thresholds. This confirmation step bolsters the security and validates the correct application of expiration thresholds. Upon receiving a request for decryption keys from a recipient with valid access, the smart contract verifies their group membership and eligibility. We have a set of recipients, denoted as R, and a set of groups, denoted as G. Each recipient, represented by $r \in R$, has a membership status, m_r, which can be binary: $m_r = 1$ for group members and $m_r = 0$ for non-members. Eligibility for decryption keys depends on this status and potential additional criteria. A smart contract handles verification: if a recipient, m_r, equals 1 and meets specified criteria, the contract distributes necessary expiration periods from EDO.

It also provides key recovery parameters to eligible recipients $r \in R$, including P_r (Lagrange polynomial formula) and C_r (specific key combinations). With these parameters (i.e., combination, rates, and expiration time), recipients can reconstruct decryption keys and access encrypted content. Expiration levels vary, resulting in different decay rates and durations. Short expiration levels (high decay rate) lead to faster key expiration, while prolonged expiration levels offer extended access (low decay rate). Following this stage, the partial decay process varies across distinct user groups, adapting to their specific requirements and access privileges. However, once the final time threshold is reached, all groups across different levels will collectively forget the random sources responsible for generating the polynomial. Consequently, this collective forgetting prevents them from recalling the key, particularly since the key is not stored; it is solely generated from the EDO to create and obscure the origin of the points. Implementing this scheme facilitates selective forgetting, allowing for managing decay rates associated with the expiration period to formalize group access. Simultaneously, it ensures complete decay occurs to promote digital oblivion.

Table 3.2: Parameters Used in The Simulation Study.

Parameters	Description	Variable Range
K	Threshold of the polynomial	$[3, 5, \ldots, 60]$
N	Total number of the polynomial points	$[10, \ldots, 120]$
λ	Decay rate	$[0.1 \ldots, 100]$
μ	Stability mean	$\in \mathbb{Z}, \quad \widehat{\mu_i} \in [-50, 50]$
m	Total number of entities	$[1, \ldots, 20]$

3.5. RESULTS

We created a prototype of our multi-level scheme with key decay on the local Ethereum blockchain[2], developed smart contracts using Solidity with Hardhat[3], interacted with the blockchain via Web3.js[4], and exposed an API using Express.js[5].

We developed a Python script for key construction and reconstruction using polynomial-based threshold cryptography to securely distribute and recover ephemeral keys. This script enforces controlled key expiration with varying decay rates, preventing unauthorized reconstruction beyond predefined thresholds. The system was tested using a YouTube dataset and pagination-supported APIs for key seed generation [101].

This seed was employed to fit a Lagrange-basis polynomial, accommodating static and exponential decay rates for efficient data expiration. Our rigorous experiments, conducted on a MacBook with a 2.3 GHz Dual-Core Intel Core *i*5 processor and 8 GB of memory, explored various parameters (listed in Table 3.2), including K, N, λ, μ, and m, to comprehensively assess their impact on system performance. Our experiments covered three key areas: (*i*) Decay sensitivity: This term evaluates how different decay fashions impact key expiration and data security, (*ii*) Computational complexity: This concept assesses the system's efficiency in managing key expiration, and (*iii*) Security analysis: This term refers to the assessment of the system's security measures.

3.5.1. DECAY SENSITIVITY

In the evaluation of decay sensitivity, we examine the effects of the decay process on key expiration in various scenarios. Our study emphasizes four fashions, namely (*i*) Time-based decay, (*ii*) Granularity decay, (*iii*) Contextual decay, and (*iv*) Adaptive decay.

TIME-BASED DECAY

Focuses on understanding how the decay process influences key expiration over time. By analyzing the decay rate and observing the expiration of keys at different time intervals using a multi-level scheme, we can evaluate the effectiveness of the time-based decay mechanism in managing key expiration. Our evaluation examines two main aspects: partial decay and complete decay.

Partial Decay. To evaluate partial decay per group, we consider a scenario where our system comprises three groups: *short, medium,* and

[2] https://github.com/trufflesuite/ganache
[3] https://github.com/NomicFoundation/hardhat
[4] https://github.com/ChainSafe/web3.js
[5] https://github.com/expressjs/express

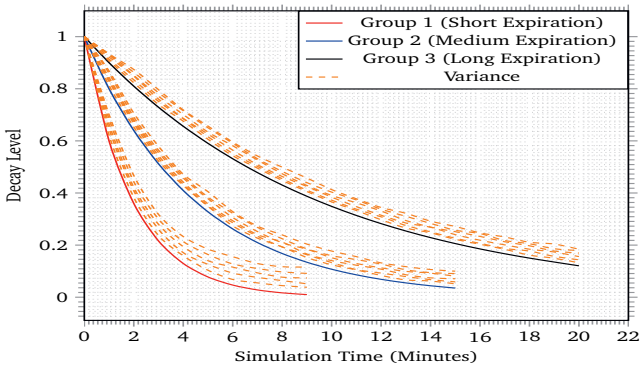


Figure 3.4: Partial Decay Evaluation For Various Groups.

long expiration thresholds. It is important to note that, for the sake of explanation, we specifically focus on these three groups (containing a random μ and m) in this instance. We track the decay rate and observe the corresponding key expiration for each group over a specific period of time. This analysis allows us to assess the sensitivity of decay by investigating how different expiration thresholds affect the decay process within each group. Figure 3.4 visually compares the decay rates and key expiration among the groups. In our experiment, the decay level represented the remaining key value using the formula: $DecayLevel = 1 - \lambda$, with λ denoting the decay rate over time. We used exponential decay functions, a common model for natural decay, where the rate is proportional to the current value. Group 1, with a short expiration and high decay rate, saw a rapid key value decline. Group 2, with a medium expiration and moderate decay rate, exhibited slower decay. Group 3, with a long expiration and low decay rate, experienced a gradual decline. This illustrates how expiration thresholds and decay rates impact each group's decay process.

The results showed decay levels over time for the three groups. Group 1 starts with an initial decay level of 1, decays to 50% in about 3 minutes, and to 90% in approximately 6 minutes with a 10-minute expiration. Group 2 starts with a similar decay level, reaches 50% in around 5 minutes, and 90% in about 12 minutes with a 15-minute expiration. Group 3, with a 20-minute expiration, takes about 7 minutes to reach 50% decay and approximately 17 minutes to hit 90%. We deliberately selected different decay rates to create variations in group expiration (Table 3.2). The plotted variance represents outcomes from multiple experiments with varying decay rates, introducing uncertainty and variability. Adding noise simulates variations that might occur when repeating experiments, with a range of variance around ± 0.1 from the original decay curves. The magnitude of added noise dictates the level

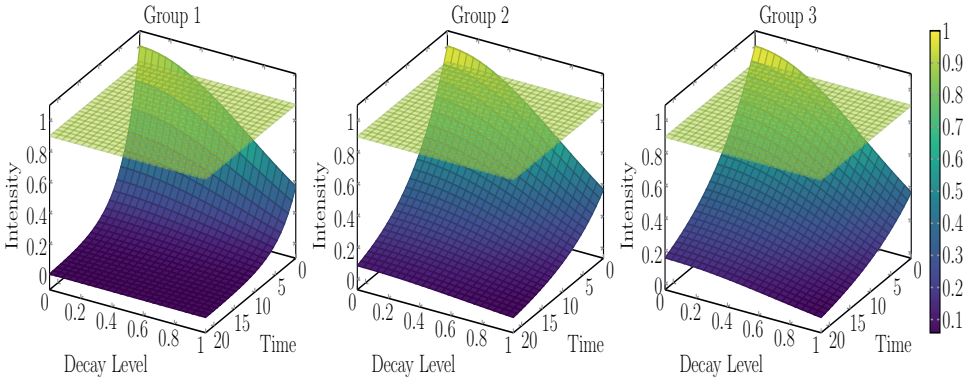


Figure 3.5: Complete Decay Evaluation Across Different Groups.

of variability, underscoring the need for robust experimental design and replication to ensure reliable results. Multiple experiments with different decay rates reveal the range of possible outcomes, assess consistency, and identify outliers or unexpected patterns.

Complete Decay. The complete decay evaluation analyzes the total decay rate and key expiration across all groups, complementing the partial group-specific analysis. Figure 3.5 illustrates the combined effect of partial and complete decay. To represent the decay process for each group, we use decay intensity functions: $I_1(t)$ for Group 1, $I_2(t)$ for Group 2, and $I_3(t)$ for Group 3. These functions capture the gradual decrease in decay level over time t for each group, expressed as $I(t) = \sum_{i=1}^{3} e^{-\lambda_i t}$. In these formulas, λ_1, λ_2, and λ_3 represent the decay rates for each group. A higher value of λ corresponds to a faster decay rate (i.e., less μ values), resulting in a more rapid decrease in the decay level over time.

Each group is represented in the experiment by a unique decay process characterized by its specific decay intensity function. These functions for the three groups are given by $\exp\left(-\frac{\lambda}{5}t\right)$, $\exp\left(-\frac{\lambda}{8}t\right)$, and $\exp\left(-\frac{\lambda}{12}t\right)$, respectively. These values are selected to represent experimental decay rates λ in three distinct groups, showcasing the varying speeds at which keys degrade over time in the experiment. Comparing these rates, Group 1 has the highest λ, indicating the fastest decay, followed by Group 2 and Group 3. Thus, Group 1 experiences the quickest partial decay, decreasing in around 10 minutes, while Group 2 and 3 decay more slowly, taking roughly 15 and 20 minutes, respectively. The top surface in each plot represents "complete decay" with a constant intensity value of 0.9 throughout the experiment. This intensity value serves as the threshold, indicating that a key is fully decayed and no longer usable for decryption or accessing information when its decay level reaches or surpasses this value.

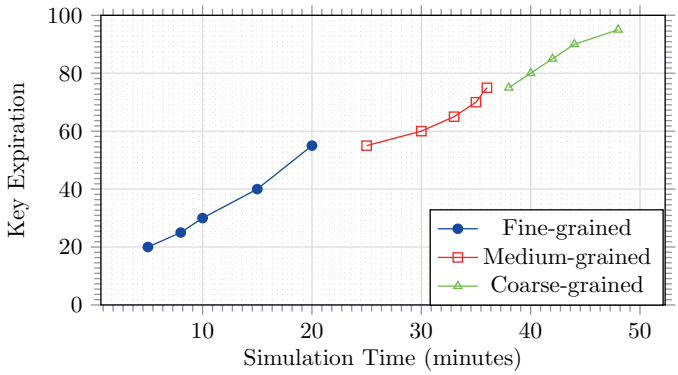


Figure 3.6: Key Expiration(%) and Granularity Levels.

The final decay time is set at 22 minutes, signifying that after this point, all keys in all groups have irreversibly decayed, highlighting the loss of associated information. These visualizations illustrate key decay from their initial values to the final level (typically represented as 0), highlighting the irreversible nature of the process.

GRANULARITY DECAY

Our research investigates key expiration across different granularity levels: fine-grained, medium-grained, and coarse-grained categorizations. We aim to understand the decay pattern by tracking key expiration rates over time using specific key combinations. Our investigation involves three distinct groups to align granularity levels and assess their characteristics.

Figure 3.6 illustrates the relationship between key expiration and granularity levels. Fine-grained keys, offering more key recovery possibilities, exhibit a slower decay rate and lower expiration percentage compared to medium-grained and coarse-grained keys. Fine-grained keys start at 20% expiration and reach 55%, while medium-grained keys begin at 55% and reach 75%, and coarse-grained keys start at 75% and progress to 95% expiration. These findings emphasize the importance of choosing the right granularity level for key categorization. More possibilities within a key combination lead to a slower decay process, offering insights for designing partial decay mechanisms that align with specific granularity requirements.

CONTEXTUAL DECAY

We also explore the impact of contextual factors on the decay process and key expiration. Contextual decay takes into account various factors that may influence the decay rate and the expiration of keys. These

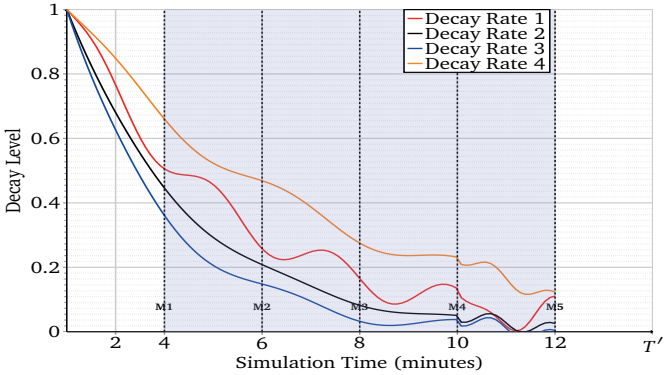


Figure 3.7: Comparison of Partial Decay Rates in Adaptive Decay.

factors can include the usage patterns of keys, the type of data associated with the keys, or the sensitivity level of the data. For example, keys associated with frequently accessed data may have a different decay rate than those associated with rarely accessed data. Similarly, keys containing highly sensitive information may expire shorter than keys with less sensitive information. In our use case, data owners can define rules and parameters for each level, enabling them to customize the decay process according to their data's unique attributes and needs (see Section 3.3.4).

ADAPTIVE DECAY

This concept refers to dynamically adjusting the decay process based on specific criteria or feedback. It enables fine-tuning decay parameters like decay rates and expiration thresholds in response to system performance or user needs. In our case, this adaptive decay empowers data owners to proactively manage data, allowing them to shorten or revoke it as needed. This flexibility is achieved by separating levels and rules, allowing for manual revocation or rule changes. Mathematically, adaptive decay can be expressed as follows:

$$\lambda = f(M) = \begin{cases} \lambda_{\text{high}} & \text{if } M > T, \\ \lambda_{\text{low}} & \text{if } M < T', \\ \lambda_{\text{default}} & \text{otherwise.} \end{cases} \tag{3.9}$$

In this formula, λ represents the decay rate and M denotes a certain measure or metric (e.g., average access frequency) that influences the adaptive decay process. The function f evaluates these parameters and determines the appropriate decay rate based on the user's preferences or system requirements. The decay rate can be dynamically adjusted by assigning different values to λ_{high}, λ_{low}, and λ_{default}. If the access metric

M exceeds a certain threshold T, indicating a higher preference for key expiration, the decay rate is set to λ_{high}. Conversely, if M falls below a threshold T', indicating a lower preference for expiration, the decay rate is set to λ_{low}. For all other cases, where M is within an acceptable range, the decay rate defaults to λ_{default}. Figure 3.7 demonstrates the scalability of the scheme with four decay rates, although it can accommodate any number. The first rate exhibits exponential decay with a constant of 0.3 and includes sinusoidal oscillations (i.e., periodic fluctuations upon M) at a frequency of 150, leading to decay fluctuations. The second rate has a higher constant of 0.4, resulting in faster decay, with minor sinusoidal fluctuations at 120 frequency. The third rate, with the highest constant of 0.5, decays rapidly and displays noticeable fluctuations due to a 100-frequency sinusoidal component. The last rate, with a constant of 0.2, decays more slowly and includes a 100-frequency sinusoidal component, causing moderate fluctuations. Between 4 and 12 minutes, metrics ($M1 \rightarrow M5$) introduce additional sinusoidal terms, further altering decay rates and enhancing complexity. Users can fine-tune parameters using the adaptive mechanism to align with their information retention goals and preferences.

3.5.2. COMPUTATIONAL COMPLEXITY

We compared the computational requirements of two forgetting schemes, multi-level and key decay, as summarized in Table 3.3. Our analysis evaluated efficiency, scalability, security, flexibility, and key management aspects, providing insights into their computational characteristics and trade-offs. Our experiments analyzed the time and space complexities of the multi-level and decay schemes. The multi-level scheme exhibited a quadratic time complexity ($O(m^2)$, e.g., 100 ms for 10 data points, 400 ms for 20). Its space complexity $O(s(m))$ grew linearly with data points (e.g., 100 KB for 10 data points, 200 KB for 20). This reflects the impact of decentralization on computational requirements. Conversely, the decay scheme demonstrated linear time

Table 3.3: A Comparison of Multi-Level and Key Decay Schemes.

Aspect	Multi-Level Scheme	Decay Scheme
Time Complexity	$O(m^2)$	$O(m)$
Space Complexity	$O(s(m))$	$O(s(\lambda))$
Security	Key Decay	Key Decay
Flexibility	High	Low
Scalability	Scalable	Fixed
Key Management	Granular Control	Limited

complexity ($O(m)$, e.g., 10 ms for 10 data points, 20 ms for 20), making it efficient with larger datasets. The decay scheme's space complexity $O(s(\lambda))$ varied with the decay rate. For instance, with a 0.5 decay rate and 5 KB storage per unit of the decay rate, it was 2.5 KB. Higher decay rates resulted in a faster reduction of required storage space. Both schemes prioritize security by gradually corrupting keys over time. The multi-level scheme provides customized expiration policies and access privileges at the group level, offering flexibility. In contrast, the decay scheme applies a uniform decay process to all keys. In terms of scalability, the multi-level scheme efficiently adapts to varying group numbers and data, accommodating expanding datasets. Conversely, the decay scheme has fixed scalability, treating all groups uniformly without the ability to scale. Finally, Key management is a strength of the multi-level scheme, granting data owners control over distinct expiration rules and access privileges for each group. However, the decay scheme lacks such key management capabilities.

3.5.3. SECURITY ANALYSIS

In this section, we assess the security of the proposed approach regarding various security goals and threats.

RETROSPECTIVE PRIVACY.

The proposed scheme introduces a novel key management and expiration method that enhances security by ensuring complete key expiration. It incorporates the concept of key decay and utilizes distinct levels with associated contracts for partial decay. This approach significantly reduces the risk of data exposure and unauthorized decryption, even if an attacker gains access to the data later.

BRUTE FORCE ATTACK.

The multi-level data forgetting scheme may face brute force attacks, where adversaries attempt to guess the key's origin. To counter this threat, the scheme employs different combinations for each level, making it more complex for attackers to determine the key's origin. Additionally, an obfuscation process conceals the key's sources, further thwarting brute-force attacks (see Section 3.3.1).

DYNAMIC EXPIRATION PERIODS.

This research extends the decay scheme to offer multiple key recovery options through partial and complete decay, enabling users to manage decay rates for diverse expiration periods. Smart contracts enhance this by enforcing rules based on user preferences, providing greater flexibility and control over data accessibility duration.

EXTENDING EXPIRATION PERIODS.

The proposed approach enables data owners to update rules and extend expiration periods by publishing new rules to smart contracts, granting flexibility according to evolving circumstances (as discussed in adaptive decay 3.5.1).

MANUAL REVOCATION.

The scheme supports manual revocation by assigning unique identifiers to user groups. Data owners can revoke access to specific groups by resetting shared parameters through smart contracts without waiting for the decay scheme to complete. Setting the parameters $P = N = 0$ (as in Section 3.5.1) prevents the reconstruction of the private key combination and the decryption of online content during the validity period.

3.6. RELATED WORK

This section summarizes recent developments in digital forgetting and online data revocation. These studies aim to address the need for secure and efficient deletion of digital information without considering audience-based expiration. Our proposed framework is based on the key decay scheme presented by Marwan et al. [56], allowing keys to exist temporarily without requiring centralized storage during their validity period. However, a notable limitation of this approach is the need for more flexibility in accommodating different user groups. Various techniques have been proposed to handle expiration periods, including flexible and fixed single-level expiration periods for all audiences. Examples include Ephemerizer [64], Vanish [65], Neuralyzer [66], EphPub [44], and Timed Revocation [67]. Another approach [81] uses smart contracts on a local Ethereum blockchain to enforce revocation conditions based on contractual agreements between cloud providers and data owners. In the realm of provable data deletion schemes, Yang et al. [82] introduced the number-rank-based Merkle hash tree (NR-MHT) for efficient data integrity auditing and dynamic data insertion while ensuring provable data deletion for the same audience. For secure deletion in IoT devices, Xiong et al. [83] proposed a key derivation encryption algorithm based on flash memory's hierarchical structure. This algorithm involves simultaneous key deletion for all users, encompassing both cipher form and key-related components when data validity ceases. The Forgits data structure strengthens online deletion by gradually dropping the lowest bits or pixels (least significant bits) from old to new data, enabling infinite retrievals by forgetting older stored data without impacting other user groups [86]. Despite significant progress in digital forgetting techniques, they still face limitations in

Table 3.4: Comparative Analysis of Research Studies: Examining Different Approaches.

Research Name	Delete Content	Reduce Exposure	Flexibility	Manual Revocation	Scalability	User-Level Expiration
Ephemerizer [64]	✓	✗	✗	✗	✗	✗
Vanish [65]	✓	✗	✗	✗	✗	✗
EphPub [44]	✓	✗	✗	✗	✗	✗
Timed Revocation [67]	✓	✗	✗	✓	✗	✗
Neuralyzer [66]	✓	✗	✓	✓	✓	✗
Towards Contractual Agreements [81]	✓	✗	✓	✓	✓	✗
Provable Data Deletion Scheme [82]	✓	✗	✓	✓	✓	✗
IoT Data Deletion Scheme [83]	✓	✗	✓	✓	✓	✗
Forgits [86]	✓	✗	✗	✗	✗	✗
Key Decay [56]	✓	✗	✓	✗	✗	✗
Proposed Scheme	✓	✓	✓	✓	✓	✓

dynamically revoking access to files across different user groups. Each reviewed technical proposal concentrates on particular functionalities within this context. In order to offer a comprehensive comparison, we thoroughly considered the key metrics within the digital forgetting domain in relation to our proposed scheme; we have summarized our findings in Table 3.4.

Delete Content. Enables content removal from the platform, giving users control over its accessibility and the option to delete it.

Reduce Exposure. Provides selective content forgetting on the platform, allowing users to control access and share it with specific audiences.

Flexibility. Enables customizable expiration periods for digital content, allowing users to specify timeframes for automatic deletion or inaccessibility, providing flexible content lifespan management.

Manual Revocation. Empowers users to revoke access to their content manually anytime. This gives them control over who can access their content and allows them to change permissions or revoke access privileges as needed.

Scalability. Pertains to the system's ability to efficiently manage a

large volume of users and data while maintaining optimal performance.
User-Level Expiration. Enables disjunctive personalized expiration periods for user content, offering individual control over accessibility and deletion timelines.

3.7. CONCLUSION

We introduced a novel multi-level data forgetting scheme using key decay techniques. The ubiquity of digital data demands solutions for audience-dependent data transience and ephemerality. Our scheme empowers data owners to manage expiration periods for different user groups, providing better content control and customization. This flexibility enhances content management and personalized forgetting, creating a comprehensive framework for optimizing content management while ensuring improved accessibility and privacy across user groups. We have implemented and evaluated a prototype, yielding promising results for our scheme's effectiveness.

4

DATA CO-OWNERSHIP DIGITAL FORGETTING

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 chapter introduces a Policy-Based Conjunctive Scheme (PBCS) that enables conjunctive decision-making for data access and collaborative data forgetting, aligning with the 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.

This chapter is based on the paper "A Policy-Based Conjunctive Scheme for Digital Forgetting of Co-Owned Data" by Darwish, M.A., Markatou, E.A., & Smaragdakis, G., which is under review.

4.1. 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 [102]. However, this evolution towards cloud storage services unveils data security vulnerabilities due to users' reduced control over their outsourced data [103–105]. These issues raise critical concerns about the robustness and reliability of data deletion strategies [59, 106–108]. 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) [60, 109–111] in 2018, and the concept of Ephemerality (i.e., Temporary existence of data after a set period) [30, 56, 112] 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 [65], EphPub [44], Ephemerizer [64], Safevanish [113], Timed Revocation [67] and Forgetting with Puzzles [45] use encryption, decentralized ephemeral storage such as DHT [91] and DNS [90] 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 [43]) and involving co-owners in deletion processes [114–117]. Previous models, such as the Disjunctive forgetting scheme [87], introduced group-specific expiration keys via smart contracts but were limited by rigid user segmentation, preventing overlapping group participation. Current approaches [45, 56, 65, 67, 118] 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 growing 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 [119] to address the limitations of previous models and to advance digital forgetting practices in cloud-based file-sharing. The scheme is well-suited for scenarios where data owners and co-owners share responsibilities and rights over 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 [120, 121]. 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 unauthorized access risks.

To accommodate overlapping group memberships, PBCS offers flexible user categorization and nuanced access control across various groups, allowing dynamic cross-group access—a key feature for collaborative environments. This reflects the scheme's conjunctive model, where members must meet collective or 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 allows PBCS to efficiently respond to evolving data relevance and security needs. As data loses relevance, deletion schedules are aligned with the keys' predefined lifespans, ensuring a smooth transition into the final stage, the "*forgetting phase*." During this phase, encryption keys degrade, following the principles of RTBF, significantly reducing the risk of data breaches. Ephemeral keys ensure

data inaccessibility over time, while co-owners initiate deletion through a semi-honest provider based on 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 following 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.

4.2. DESIGN GOALS

4.2.1. PROBLEM STATEMENT

Can we design a collaborative digital management system that dynamically handles file sharing, ownership rights, and data forgetting while ensuring GDPR compliance (Article 17 & 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 [43, 62, 122]. Conventional systems are often inadequate for scenarios requiring nuanced data control and privacy measures [10, 123–129].

In collaborative environments like writing clubs, managing copyright terms dynamically among multiple authors presents 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 [130, 131]. For example, members' contributions to a collective work in a writing club are subject to specific copyright terms that evolve over time [132]. 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.

4.2.2. 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 of each data object, which includes access and expiration parameters to control the data lifecycle.
***Co-Owners* (COs):** The COs are authorized to access and delete co-owned data objects 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 protocol requirements due to legal and operational incentives but is curious to infer unauthorized information from stored data, reflecting realistic scenarios.
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 §4.4.4.

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 SSSS, 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-owners' 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.

4.3.1. Definitions

Table 4.1 lists the notations used in the study.

Definition 1. *A symmetric encryption scheme consists of the following components:*

Table 4.1: 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 [133].

Key Generation: *A probabilistic algorithm* $KeyGen(\kappa) \rightarrow k_s$ *that, given a security parameter* κ*, generates a symmetric key* k_s*. This step is based on Definition 4.*
Encryption: *A deterministic function* $Enc_{k_s} : \{0,1\}^* \rightarrow \{0,1\}^*$*, that takes a plaintext* m *and outputs a ciphertext* c*.*
Decryption: *A deterministic function* $Dec_{k_s} : \{0,1\}^* \rightarrow \{0,1\}^*$*, that takes a ciphertext* c *and outputs the plaintext* m*. Formally,* $c = Enc_{k_s}(m)$ *and* $m = Dec_{k_s}(c)$*, such that* $Dec_{k_s}(Enc_{k_s}(m)) = m$*.*

Definition 2. *An asymmetric encryption scheme consists of the following components:*
Key Generation: *A probabilistic algorithm* $KeyGen(\kappa) \rightarrow (k_{pub}, k_{pri})$ *that, given a security parameter* κ*, generates a public key* k_{pub} *and a private key* k_{pri}*.*
Encryption: *A deterministic function* $Enc_{k_{pub}} : \{0,1\}^* \rightarrow \{0,1\}^*$ *that takes a plaintext* m *and outputs a ciphertext* c*.*
Decryption: *A deterministic function* $Dec_{k_{pri}} : \{0,1\}^* \rightarrow \{0,1\}^*$ *that takes a ciphertext* c *and outputs the plaintext* m*. Formally,* $c = Enc_{k_{pub}}(m)$ *and* $m = Dec_{k_{pri}}(c)$*, such that* $Dec_{k_{pri}}(Enc_{k_{pub}}(m)) = m$*.*

Definition 3. *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.*

Definition 4. *Our KSS extends SSSS [70] 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 $S_{entropy}$:*

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

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:*

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

Definition 5. *The decay function D ensures point degradation over time by combining entropy-driven randomness and policy-enforced control. Given $\tau \in \mathbb{R}^+$ (time), $\lambda_{passive}$ (entropy-based decay), λ_{active} (owner-defined decay), and data points $(x_i, y_i) \in \mathbb{F}_q$:*

$$D(\tau, \lambda_{passive}, \lambda_{active}, P_{policy}, (x_i, y_i)) \rightarrow (x_i', y_i'),$$

where x_i', y_i' evolve dynamically based on $S_{entropy}$, introducing randomness from external sources to ensure share unpredictability. If $\lambda_{passive}$ decay is insufficient, P_{policy} enforces controlled deterioration through λ_{active}, ensuring compliance with RTBF. Decay guarantees key irretrievability when valid shares drop below the threshold θ.

Definition 6. *Policy P specifies permissions and conditions for data access based on roles and actions. The policy includes allowed actions, effects, resources, and conditions. The policy (d) can be described as follows:*

$$Input = \{Statement, Effect, Action, Resource, Condition\},$$
$$Output = \{Policy\ for\ EDO\}.$$

Definition 7. *Our scheme $PBCS = (Upload, Preprocessing, Access\ Control, Download, Deletion, Deletion\ Validation)$ 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.*

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

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.

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

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

$$Input = \{Votes_{acc}\}, Output = \{Access\ decision\}$$

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

$$Input = \{SymEnc_{Shares(d)}(d), AsymEnc(Shares(d)), m, \theta\},$$
$$Output = d$$

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

$$Input = \{Votes_{del}\}, Output = \{Deletion\ decision\}$$

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

$$Input = \{Shares(d), Votes_{del}\}, Output = \{Result\ verification\}$$

4.3.2. SECURITY DEFINITIONS

Correctness of Protocol Correctness in the PBCS protocol ensures that cryptographic operations align with Definition 7 and Definition 6. The protocol guarantees that downloaded data matches the original upload via consistent key shares (Definition 4), while access control and deletion strictly follow policy rules.

Definition 8. *A scheme PBCS = (Upload, Preprocessing, Access Control, Download, Deletion, Deletion Validation) is correct if the conditions hold:*

1. *The d retrieved through the download matches the original upload:*

$$Download(Upload(d)[:2], m, \theta) = d.$$

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 threshold $\theta \in \mathbb{F}_q$, at $\tau \in \mathbb{R}^+$,*

the encryption polynomial $E(x)$, modified by a decay function $f(t, \lambda)$, satisfies:

$$E(x, f(t, \lambda)) < \theta \quad \forall x,$$

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 The protocol execution reveals only what the ideal model permits: 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 9. *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 $\text{Deletion}_{validation}$.*
- *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:

$$\Pr[A(V_{real}) \neq A(V_{sim})] \leq \epsilon,$$

where ϵ is negligible of the security parameter. Thus, PBCS is Λ-secure.

4.3.3. 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 [122, 134, 135]. The scheme dynamically applies expiration and access control policies supporting overlapping memberships, facilitating data forgetting among diverse groups.

Lagrange-Basis Polynomials: Lagrange polynomials [69, 136] play a crucial role in cryptographic functions, enabling interpolation to form a polynomial through specified points. Formulated as:

$$L(x) = \sum_{j=0}^{n-1} y_j \ell_j(x) \quad \Rightarrow \quad \ell_j(x) = \prod_{\substack{0 \le m \le n-1 \\ m \neq j}} \frac{x - x_m}{x_j - x_m}.$$

Where $L(x)$ represents the Lagrange interpolation polynomial and $\ell_j(x)$ denotes the Lagrange basis polynomial, both crucial for enabling cryptographic 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 §4.4.1). 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 $\text{Decay} = 1 - \lambda$, dictating the deterioration rate.

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 under the owner's policies, it ensures flexible and responsive deletion decisions, adapting to user needs.

4.3.4. SYSTEM ARCHITECTURE

As outlined in Definition 7 and illustrated in Figure 4.1, 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 in 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 via asymmetric encryption, permitting only authorized co-owners to reconstruct the key during the downloading phase and hide the key from the provider. Users are categorized into groups established by the owner, dictating a precise policy framework via shares tied to Lagrange polynomial points. The scheme dynamically adjusts to group dynamics with flexible policies that accommodate real-time threshold

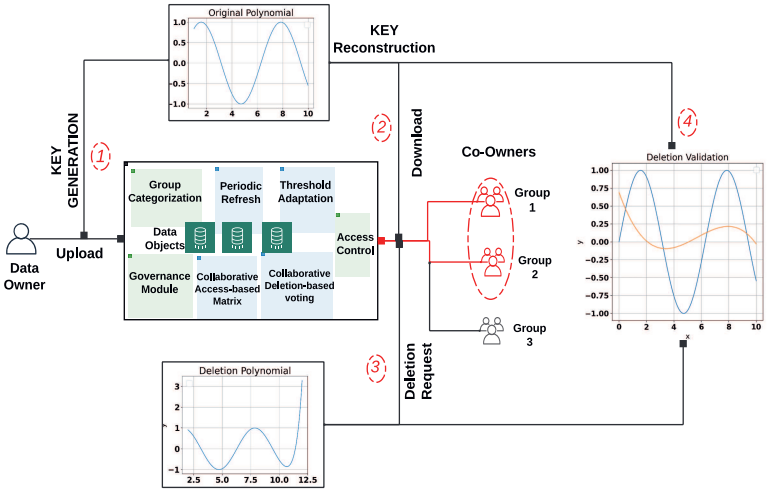


Figure 4.1: Comprehensive Architecture of The PBCS.

(i.e., minimum valid shares for key) changes 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 §4.3.3) 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 the flexibility in access, 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 §4.3.3). 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) compliance is achieved

as the system ensures data irretrievability through deletion by CP and autonomous key decay, safeguarding privacy rights. This mechanism focuses on private verifiability between DO and COs in the collaborative framework.

4.4. SCHEME DESCRIPTION

We focus on (*i*) Conjunctive Key Decay Scheme, (*ii*) Core Components, (*iii*) Formal Analysis, and (*iv*) Adversarial Models Analysis.

4.4.1. CONJUNCTIVE KEY DECAY SCHEME

The proposed PBCS introduces a hybrid key decay mechanism that combines entropy-driven passive degradation with policy-enforced active expiration based on 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 entropy sources (e.g., cryptocurrency prices, stock trends), ensuring continuous variation. Each access recomputes and hashes values, maintaining unpredictability even if some sources fail. Past shares remain unrecoverable due to hashing and the absence of real-time entropy, making reversal computationally infeasible:

$$x_i' = H(D(x_i, \lambda_{\text{passive}}, S_{\text{entropy}}, \tau)),$$

$$y_i' = H(D(y_i, \lambda_{\text{passive}}, S_{\text{entropy}}, \tau)).$$

Ensuring continuous, unpredictable decay and preventing static data persistence on the provider. To stabilize entropy-driven randomness, decay progression is controlled by averaging entropy values per share. 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 $K_y(t_i)$ represent the entropy-driven values for each share's coordinates. Decay occurs only when the average entropy across shares drops sufficiently. New entropy-driven points stabilize the key if needed, preventing premature corruption while maintaining policy control and attack resistance.

Polynomial Transformation Analysis. To further prevent key inference, polynomial transformations obscure structure and increase complexity:

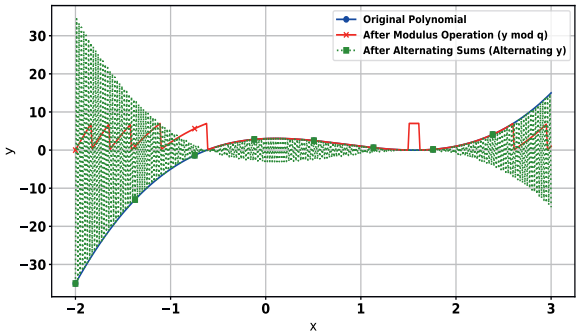


Figure 4.2: Polynomial Transformation: Modulus and Alternating Sums Functions.

- **Modulus Operation:** 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.
- **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 $P''(x) = a_0 - a_1 x + a_2 x^2 - a_3 x^3 + \ldots$. This transformation can significantly alter the polynomial's zero points and overall algebraic structure, increasing the complexity of solving it algebraically.

These transformations, as illustrated in Fig. 4.2, bolster entropy-driven decay by complicating algebraic attacks, reinforcing unpredictability, and secure forgetting.

Active Decay and Policy-Defined Expiry. When passive entropy changes are insufficient, policy-driven active decay enforces key 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.

Conjunctive Threshold for Access. Decryption requires a sufficient number of valid shares, governed by either individual thresholds θ_i and a collaborative threshold θ_{collab} (in phase 13):

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

Key irretrievability occurs when: $K(t) < \theta_{\text{collab}}$, where $K(t) = \prod_{i=1}^{n}(w_i \cdot K_i(t))^{\alpha_i}$, with w_i as the weight of co-owner i and α_i controlling share sensitivity, ensuring access is revoked when shares collectively decay beyond recovery.

4.4.2. 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, 66, and Article 17 [60, 109–111]. It manages secure data transitions across uploading, downloading, deleting, and forgetting phases (Definition 7).

Algorithm 2: 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), {AsymEnc(k_{pub}, data_points)}$_{k_{pub} \in PKs}$, policy(d)

Algorithm 3: 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, $data\_points$

UPLOADING PHASE (ALGORITHM 2)

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, Alg. 3 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. Consider a Lagrange polynomial $L(x)$ constructed from a random set of data points $\{(x_i, y_i)\} \in \mathbb{F}_q$. The polynomial is defined as $L(x) = \sum_{j=0}^{n} y_j \ell_j(x)$, where $\ell_j(x)$ are the Lagrange basis polynomials, and n the shares number. We evaluate the polynomial at zero: $L(0) = \sum_{j=0}^{n} y_j \ell_j(0)$. This value, $L(0)$, is then used as the input seed for a cryptographic hash function H: $K = H(L(0))$. The 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 [137]. 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.

Algorithm 4: Threshold Adaptation

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

1 $L_x \leftarrow$ ConstructLagrangePolynomial(*data_points*)
2 Generate $\theta\_new - 1$ random data_points $(x_i, y_i) \in \mathbb{F}_q$
3 seed $\leftarrow$ EvaluateAtZero(L_x)
4 *Updated_data_points* $\leftarrow$ Generated points+$(0, seed)$
5 **return** *Updated_data_points*

Lemma 2 (Dynamic Threshold Adaptation)**.** *The data owner can run Algorithm 4 to generate shares with a new threshold θ_{new} for the reconstruction of the same key k used in the encryption process.*

Proof. The polynomial defines the initial key seed setup:

$$f^0(x) = s + \sum_{i=1}^{\theta_{init}-1} a_i x^i,$$

over a finite field $\mathbb{F}_q$, where s is the secret and θ_{init} is the initial threshold for key reconstruction. To modify the threshold from θ_{init} to θ_{new}, we generate $\theta_{new} - 1$ new random data points $(x_i, y_i) \in \mathbb{F}_q$.

The updated Lagrange polynomial $L(x)$ is then constructed using the original and new data points, ensuring consistency with the new threshold requirements. The key seed, which aligns with the polynomial evaluated at zero, is calculated as follows:

$$L(0) = \sum_{j=0}^{\theta_{new}-1} y_j \ell_j(0),$$

where $\ell_j(x)$ represents the Lagrange basis polynomials:

$$\ell_j(x) = \prod_{\substack{k=0 \\ k \neq j}}^{\theta_{new}-1} \frac{x - x_k}{x_j - x_k}.$$

The revised set of data points thus includes both the newly generated points and the seed:

$$\text{Updated\_data\_points} = \{(x_i, y_i)\}_{i=1}^{\theta_{new}-1} \cup (0, L(0)).$$

This construction guarantees that the secret s, and consequently the symmetric key k, remains reconstructible provided at least θ_{new} unique shares are available, thereby adapting the scheme to changing access requirements while preserving security and integrity.

□

Lemma 3 (Periodic Refresh)**.** *Every refresh time t timesteps, Algorithm 5 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, determined by the Cloud Provider (CP), a renewal polynomial:

$$R^t(x) = \sum_{i=1}^{t-1} R_i^t x^i,$$

Algorithm 5: 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*

is generated with the constraint that $R^t(0) = 0$ to ensure the original secret remains unchanged. This design guarantees that the secret's foundational integrity is preserved throughout the refresh process.

In this renewal process, instead of relying on the existing shares, $n-1$ new random data points $(x_i, y_i) \in \mathbb{F}_q$ are generated, which serve as the basis for updating the shares. The refreshed shares are then calculated as:

$$x_i^{new} = R^t(i).$$

The refresh mechanism operates within a validity window, where t represents the current time interval and t_{max} is the expiration threshold. If $t < t_{max}$, new random data points are continuously generated and integrated into the system. Upon reaching t_{max}, the periodic refresh halts, after which the data points naturally decay according to a decay factor λ, as defined in Definition 5.

This decay process gradually reduces the data points' effectiveness, ensuring that, over time, the symmetric key k degrades and becomes irrecoverable. This scheduled degradation aligns with the system's requirements for key expiration and enforces the principle of data ephemerality by embedding decay directly into the cryptographic framework.

□

The CP is the key for implementing these two attributes of PBCS. The data points representing the secret shares stay on the CP and are not distributed to the COs before the download phase is due. This ensures that old shares can't 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.

Algorithm 6: Group-specific Public-Key Encryption for Polynomial Coefficients

Input: coefficients, publicKey
Output: Encrypted coefficients

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

Lemma 4. *Algorithm 6 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 employed in generating the symmetric encryption key k, where $c_i \in \mathbb{F}_q$ are the polynomial coefficients within the finite field $\mathbb{F}_q$. These coefficients, $\{c_0, c_1, \ldots, c_n\}$, are essential to reconstructing k but require secure distribution to prevent unauthorized access. The coefficients are encrypted using asymmetric key encryption specific to each group to achieve this.

For a group with a public key pub, the encryption process for each coefficient c_i is defined as:

$$c_i^{enc} = \text{Enc}_{\text{pub}}(c_i),$$

where $\text{Enc}_{\text{pub}}(\cdot)$ denotes the asymmetric encryption function using the group's public key. This encrypted form of the coefficients ensures that only group members possessing the corresponding private key can access the necessary values to reconstruct the polynomial $L(x)$.

Upon decryption, each group member utilizes the private key priv associated with pub to decrypt the encrypted coefficients:

$$c_i = \text{Dec}_{\text{priv}}(c_i^{enc}),$$

where $\text{Dec}_{\text{priv}}(\cdot)$ represents the decryption function specific to the group's private key. This operation is computationally efficient within a cloud infrastructure as the encryption and decryption for all coefficients has a complexity of $O(N)$, where N denotes the number of groups participating in the process.

The security of this approach relies on the strength of the asymmetric encryption algorithm employed. Given a sufficiently secure encryption scheme, the probability of an adversary successfully decrypting the coefficients without the correct private key is negligible. This is based on the cryptographic assumption that reversing the encryption without

knowledge of the private key is infeasible within a polynomial time, maintaining the confidentiality of k.

In conclusion, public-key encryption provides a robust mechanism for distributing polynomial coefficients securely, ensuring that only authorized group members can reconstruct the symmetric key necessary for data access. This process effectively supports the security requirements of the conjunctive scheme by limiting access to essential key components based on group-specific authorization.

□

Algorithm 7: 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 $Votes_{acc}, Votes_{del}, Val_{initial}$

PREPROCESSING PHASE (ALGORITHM 7)

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 are established ($Votes_{del}$). 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.

DOWNLOAD PHASE (ALGORITHMS 8 AND 10)

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 upon meeting policy-defined thresholds.

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 8 outputs 1 if the group is authorized to access d and 0 otherwise.*

Algorithm 8: Access-Controlled Download

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

1 $L_{access} \leftarrow ConstructLagrangePolynomial(V_{acc})$
2 **return** $L_{access} \equiv L_{acc}$

Proof. The set $G = \{(ID_i, Sig_i)\}_{i=1}^{n}$, where each tuple contains ID_i, a unique identifier for the i^{th} group, and Sig_i, a cryptographic signature generated using the group's asymmetric private key. Here, $ID_i \in \mathbb{Z}$ and $Sig_i \in \mathbb{Z}_N$, where $\mathbb{Z}_N$ denotes the group under modulo N arithmetic, ensuring compatibility with cryptographic operations. Additionally, let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be a secure cryptographic hash function, where $\mathbb{Z}_q$ is a finite field of prime order q, ensuring that hashed outputs are securely mapped within this finite domain.

To enforce access control, we construct the polynomial $L_{access}(x)$, which interpolates points based on the authorized user inputs. Define $L_{access}(x)$ as:

$$L_{access}(x) = \sum_{i=1}^{n} y_i \cdot \ell_i(x) \mod q,$$

where each y_i is derived as $y_i = H(ID_i || Sig_i \oplus \text{nonce}[128])$. The nonce serves as a 128-bit seed, introducing randomness and ensuring freshness for each access request. Here, $\ell_i(x)$ denotes the i^{th} Lagrange basis polynomial, formulated as:

$$\ell_i(x) = \prod_{\substack{1 \le j \le n \\ j \neq i}} \frac{x - x_j}{x_i - x_j} \mod q.$$

To verify access rights, the requesting group provides its ID and Sig. From these, a verification point $(x_i, \hat{y}_i)$ is derived, where:

$$\hat{y}_i = H(ID \,||\, Sig \oplus \text{nonce}[128]).$$

The system then evaluates $L_{access}(x)$ at $x = x_i$ and compares it to $\hat{y}_i$:

$$L_{access}(x_i) \equiv \hat{y}_i \mod q.$$

If equality holds, the user's credentials are confirmed, and access is granted. If not, access is denied, safeguarding against unauthorized use.

To ensure the scheme's integrity, it requires a minimum threshold θ of valid points (x_i, y_i) for constructing a unique Lagrange polynomial $L_{access}(x)$. Should the number of valid points fall below θ, $L_{access}(x)$ cannot be correctly reconstructed, thereby denying access.

The complexity of constructing and evaluating $L_{access}(x)$ is $O(N^2)$, where N denotes the number of groups involved. The scheme's security relies on the cryptographic strength of the hash function H and the nonce, making it computationally infeasible to forge valid verification points $(x_i, \hat{y}_i)$ without the legitimate ID and Sig. This preserves the robustness and security of the access control mechanism in the face of potential adversarial attempts (Refer to Algorithm 9 for the complete procedure).

□

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 Alg. 10.

Decryption Process Leveraging Public-Key Encryption and Group-Specific Access. Upon successful decryption using the group-specific asymmetric private key, the original polynomial $L_{enc}(x)$ 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.

Lemma 6. *Algorithm 11 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 $policy(d)$.

Algorithm 9: 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

Algorithm 10: 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 Alg. 12
$d \leftarrow SymcDec(SymEnc(d, k), k)$
return d

Algorithm 11: Group-specific Public-Key Decryption for Polynomial Coefficients

Input: encrypted_coefficients, privateKey
Output: Decrypted coefficients

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

$$
\begin{array}{c} \\ User_1 \\ User_2 \\ \\ User_n \end{array}
\begin{array}{c}
\begin{array}{cccc} Group_1 & Group_2 & \dots & Group_n \end{array} \\
\left[\begin{array}{cccc} m_{11} & m_{12} & \cdots & m_{1n} \\ m_{21} & m_{22} & \cdots & m_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{array}\right]
\end{array}
\begin{array}{c} V \\ \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{array}
$$

Figure 4.3: Group Membership Matrix m.

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: $V = \{\nu_1, \nu_2, ..., \nu_n\}$, where $\nu_i \in \{0, 1\}$, as illustrated in Figure 4.3.

Algorithm 12: Symmetric Key Reconstruction Type

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

Type ← *singular*
if *there exists a user i, such that* *ParticipationBased_Matrix_Approach*$[m, i] = 1$ **then**
  Type ← *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 Alg. 13)

Algorithm 13: KSS.ReconstructKey

Input: datapoints *data_points*
Output: Key k

L_x ← ConstructLagrangePolynomial(*data_points*)
seed ← EvaluateAtZero(L_x)
key ← H(seed)
return key k

Lemma 7 (Decryption Polynomial Generation)**.** *Given participation matrix m, data points shares, and threshold θ, Algorithm 12 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 the

decryption follows a singular or collaborative polynomial based on group participation and threshold requirements. For any member i of group G with unique identifiers $(x_i, y_i) \in \mathbb{F}_q$, the singular Lagrange polynomial $L_{singular}(x)$ is adapted to incorporate multiple thresholds θ_i, ensuring independent decryption when each threshold is met. The polynomial is defined as:

$$L_{singular}(x) = \sum_{i \in G} y_i \cdot \ell_i(x),$$

where $\ell_i(x)$ is the Lagrange basis polynomial calculated as: $\ell_i(x) = \prod_{\substack{j \in G \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$. Each group G reconstructs the polynomial independently when its unique threshold θ_i is satisfied. To ensure collective access, the decryption follows the principle that multiple groups must independently meet their thresholds conjunctively, ensuring access only when all groups' individual thresholds are satisfied. This secures the decryption process by requiring that groups operate independently but conjunctively meet their access criteria. For a set of users $\{U\}$ from different interconnected groups with overlapping IDs (i.e., a user belonging to multiple groups), the collaborative Lagrange polynomial $L_{collab}(x)$ is constructed to ensure secure collective access:

$$L_{collab}(x) = \sum_{i \in U} y_i \cdot \ell_{U_i}(x),$$

where $\ell_{U_i}(x)$ is calculated as $\ell_{U_i}(x) = \prod_{\substack{j \in U \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$. This ensures that each user contributes uniquely to the polynomial, preserving integrity and preventing unauthorized access. The security relies on the difficulty of interpolation with fewer than θ shares, where the error is bounded by the Lagrange remainder, ensuring deviation from the correct polynomial [138]:

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

The complexity of constructing the Lagrange polynomial is $O(N^2)$, where N refers to the number of groups. The probability of reconstruction without θ shares is negligible (i.e., $\approx \frac{1}{|\mathbb{F}_q|}$). □

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

Lemma 8. *During the validity period, Algorithm 10 returns data object d given the encrypted data object d′, shares, participation matrix m, and threshold θ, and has $O(N^2)$ complexity.*

Proof. To securely reconstruct the symmetric key k, we begin by decrypting the set of encrypted data points using the group-specific private key, k_{pri_g}. Each *data_point* corresponds to a fragment of the key structure, encrypted with the public key associated with the group. These data points, once decrypted, form the essential inputs for the Lagrange interpolation necessary to reconstitute the symmetric key.

The decrypted data points are then processed using Lagrange polynomial interpolation, leveraging the inherent properties of polynomial interpolation over a finite field $\mathbb{F}_q$. This process ensures that k, the symmetric key, can only be reconstructed when a sufficient number of data points are present, as dictated by the predefined threshold θ. Specifically, the polynomial $L(x)$ is constructed such that it passes through at least θ unique points (x_i, y_i), ensuring resilience against unauthorized key reconstruction attempts.

Mathematically, the reconstruction polynomial $L(x)$ is defined as follows:

$$L(x) = \sum_{i=1}^{n} y_i \cdot \ell_i(x),$$

where y_i are the values obtained from decrypted data points and $\ell_i(x)$ denotes the Lagrange basis polynomial for the i^{th} point:

$$\ell_i(x) = \prod_{\substack{1 \le j \le n \\ j \neq i}} \frac{x - x_j}{x_i - x_j}.$$

Once $L(x)$ is determined, the symmetric key k can be directly derived by evaluating $L(x)$ at $x = 0$, assuming $L(0)$ encapsulates the cryptographic seed necessary for k. If $L(x)$ correctly interpolates the designated threshold points, $L(0)$ will yield the key k accurately.

After reconstructing k, we proceed to decrypt the data object d using k under symmetric encryption, allowing local decryption independent of cloud provider involvement. This process ensures compliance with data access control policies within the designated validity period, during which data points are considered intact and have not yet succumbed to decay.

The computational complexity of this reconstruction is primarily due to the polynomial interpolation, which operates at $O(N^2)$, where N is the number of groups contributing to the reconstruction. This ensures that the process remains efficient yet robust, even as N scales, accommodating the multi-group, conjunctive framework of the PBCS.

By adhering to this structured approach, we ensure the integrity and exclusivity of the symmetric key k, maintaining strict access control

based on the availability and authenticity of the requisite data points. This ensures that only authorized groups, collectively fulfilling the threshold requirements, can access the decrypted data.

□

We expand upon what happens after the validity (i.e., set by the policy) period below in the *Forgetting Phase* in §5.

DELETION PHASE (ALGORITHM 14)

In compliance 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.

Algorithm 14: Deletion

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

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

Lemma 9 (Data Object Deletion)**.** *Given* $Votes_{del}$ *created according to policy*(d) *of data* d*, Algorithm 14 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. To construct the deletion polynomial, we adopt flexible decision-making mechanisms grounded in *majority voting* and *veto rights*, allowing fine-grained control over co-owned data deletion (see Figure 4.4). Various majority voting techniques can be applied depending on system requirements:

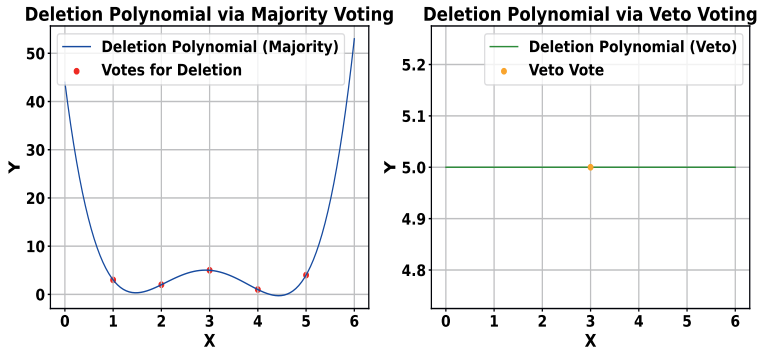


Figure 4.4: Deletion Polynomial-Based Voting Mechanisms.

- **Simple Majority:** A decision is approved if it receives more than half of the votes:

$$\text{Victory} = \begin{cases} 1, & \text{if } \frac{v}{N} > \frac{1}{2} \\ 0, & \text{otherwise,} \end{cases}$$

where v is the number of votes in favor and N is the total number of votes.

- **Absolute Majority:** A predefined quota (e.g., two-thirds) must be met or exceeded:

$$\text{Victory} = \begin{cases} 1, & \text{if } \frac{v}{N} \geq q \\ 0, & \text{otherwise,} \end{cases}$$

where q is the quorum fraction, such as $\frac{2}{3}$ or $\frac{3}{4}$.

- **Relative Majority (Plurality):** The option with the highest number of votes wins:

$$\text{Victory} = \begin{cases} 1, & \text{if } v = \max(\mathbf{v}) \\ 0, & \text{otherwise,} \end{cases}$$

where $\mathbf{v}$ is the vector of votes across all options.

- **Weighted Majority:** Votes carry different weights, with the outcome determined by the sum of weighted votes:

$$\text{Victory} = \begin{cases} 1, & \text{if } \sum_{i=1}^{N} w_i \cdot v_i > \frac{1}{2} \sum_{i=1}^{N} w_i \\ 0, & \text{otherwise,} \end{cases}$$

where w_i is the weight and v_i the vote of the i-th participant.

- **Threshold Majority:** A decision is accepted if it meets a fixed threshold t (e.g., 60%):

$$\text{Victory} = \begin{cases} 1, & \text{if } \frac{v}{N} \geq t \\ 0, & \text{otherwise,} \end{cases}$$

where t is a predefined threshold level.

These mechanisms allow the system to balance fairness, inclusiveness, and control, depending on the context and stakeholder roles. The framework also supports veto rights for high-sensitivity decisions, ensuring that critical stakeholders retain final control over deletion events.

Lemma 10 (Deletion Polynomial-based Majority Voting)**.** *Algorithm 15 constructs a deletion polynomial $D(x)$ if $Votes_{del}$ meet the majority threshold with $O(N^2)$ complexity.*

Algorithm 15: 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
```

Proof. In the Polynomial-Based Majority Voting mechanism, each co-owner i participates by casting a binary vote v_i for data deletion, where $v_i \in \{0, 1\}$. Here, 1 represents a vote in favor of deletion, while 0 represents a vote against it. Given n co-owners, these votes are aggregated into a set $Votes_{\text{del}} = \{v_1, v_2, \ldots, v_n\}$. A deletion is approved if the total number of affirmative votes meets or exceeds the majority threshold $T_m = \lceil \frac{n}{2} \rceil$.

To enforce this process within a polynomial framework, we construct a deletion polynomial $D(x)$ only when the affirmative votes reach the required threshold. The polynomial activation function is defined as follows:

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

where (x_i, y_i) are coordinate pairs associated with each $v_i = 1$, contributing to the polynomial's terms. This construction leverages Lagrange interpolation for the affirmative votes, forming a polynomial that encapsulates the decision based on majority consensus.

If $\sum_{i=1}^{n} v_i$ satisfies the threshold T_m, the polynomial $D(x)$ is constructed and activated. If not, it remains undefined, thus preventing unauthorized deletion actions. This structure facilitates a collective decision and ensures cryptographic rigor by enabling polynomial validation. Consequently, $D(x)$ inherently encapsulates the outcome of a collective majority, ensuring that data deletion adheres to agreed governance policies within the collaborative framework. □

Lemma 11 (Veto Voting Right)**.** *Algorithm 16 constructs a deletion polynomial $D(x)$ if the hashed secret $H(s_v)$ matches the input and null otherwise with $O(1)$ complexity.*

Algorithm 16: Veto-Based Deletion Polynomial

Input: Hashed_Secret, Veto_Point
Output: Deletion polynomial $D(x)$ or null

```
if ValidateHash(Hashed_Secret) then
    D(x) ← (vetoPoint · LagrangeBasis(0, [vetoPoint])) mod q
    return D(x)
end
return null
```

Proof. In scenarios requiring additional security, a single co-owner can be granted veto rights and empowered to halt the deletion process independently. This is realized by incorporating a hashed secret s_v, linked to a designated zero point of the polynomial $f(0)$. Here, $H(s_v)$ represents the hash of this secret, which effectively acts as a unique identifier for the veto.

The deletion polynomial $D(x)$ is augmented with a veto component as follows:

$$D_{veto}(x) = D(x) + H(s_v) \cdot \delta(x) \mod q,$$

where $\delta(x)$ is a Kronecker delta function, defined such that $\delta(x) = 1$ for $x = 0$ and 0 otherwise. This function ensures that the veto power directly influences $D(x)$ only at the polynomial's origin. The addition of $H(s_v)$ modulates $D(x)$ such that the veto is triggered when $H(s_v)$ is valid and $x = 0$, instantly meeting the deletion criteria.

By utilizing the Kronecker delta function [139], $H(s_v)$ directly embeds the veto influence within $D(x)$ without requiring majority consensus. This approach ensures that the right to veto can be exercised as direct, cryptographic control over data deletion, independent of other votes.

It highlights the flexibility of the polynomial-based system to integrate distinct governance controls, facilitating both collective and individual decision-making within the deletion process. □

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, ensuring deletion does not proceed. 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.

FORGETTING PHASE

Following key decay, COs may invoke their RTBF rights by requesting deletion validation from the CP, which confirms secure data erasure. Assuming an encryption polynomial $E(x)$ influenced by both entropy-driven passive decay and policy-enforced decay, the key's effectiveness diminishes over time. Passive decay, governed by S_{entropy}, introduces randomness from external sources, while active decay, controlled by P_{policy}, ensures expiration when passive entropy alone is insufficient. The decay rates, set by the DO, dictate how rapidly key effectiveness declines. When effectiveness drops below a threshold θ, decryption becomes unfeasible:

$$E(x, f_{\text{passive}}(t), f_{\text{active}}(t)) < \theta \Rightarrow \{\text{Forgetting}\}.$$

In the conjunctive decay model, access requires meeting either individual thresholds θ_i or a collective threshold θ_{collab}. Decay occurs when weighted shares across all groups fall below their respective thresholds:

$$\bigcap_{i=1}^{n} (w_i \cdot K_i(t))^{\alpha_i} < \theta_i,$$

or when the collective product decays beneath θ_{collab}:

$$K(t) = \prod_{i=1}^{n} (w_i \cdot K_i(t))^{\alpha_i} < \theta_{\text{collab}},$$

where α_i determines each co-owners share decay sensitivity. This conjunctive threshold ensures collaborative access control, while entropy-driven changes prevent static archival retrieval. 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:

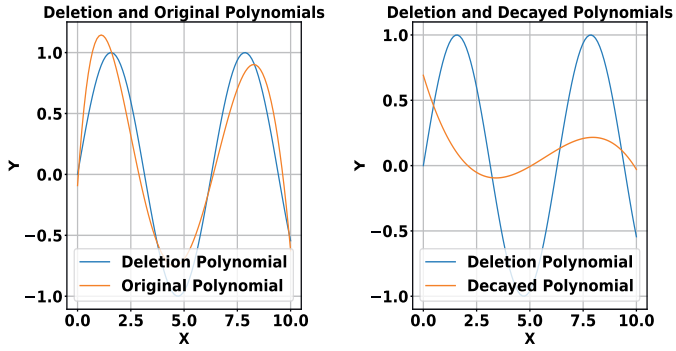


Figure 4.5: Deletion Validation: Intersection Between Original and Deletion Polynomials Within Different Periods.

$$\lim_{t\to\infty} E(x, f_{\text{passive}}(t), f_{\text{active}}(t)) = 0.$$

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, 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 4.5).

Lemma 12 (Deletion Validation)**.** *If data object d has been deleted, Alg. 17 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. To validate the secure deletion process, we define two polynomials: $E(x)$ for encryption/decryption and $D(x)$ for data deletion. These polynomials have degrees m and n respectively, so that $\deg(E) = m$ and $\deg(D) = n$. The deletion validation operates by evaluating the difference polynomial $R(x) = E(x) - D(x)$, where $\deg(R) = \max\{m, n\}$, assuming $E(x) \neq D(x)$ and that they have distinct leading coefficients.

Algorithm 17: 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 $f(t)$
$m \leftarrow \deg(E), \quad n \leftarrow \deg(D), \quad R(x) \leftarrow 0$
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

To confirm deletion, we examine the roots of $R(x)$, which indicate intersections between $E(x)$ and $D(x)$. The polynomial $R(x)$ is constructed as:

$$R(x) = \sum_{k=0}^{\max\{m,n\}} r_k x^k,$$

where the coefficients r_k are determined from the differences between the corresponding terms in $E(x)$ and $D(x)$. We define the set $S = \{x \in \mathbb{C} \mid R(x) = 0\}$, representing all points where $E(x)$ and $D(x)$ intersect.

For the deletion validation to succeed, at least one root in S must be found within a specified domain. The existence of such roots $\{x_1, x_2, \ldots, x_k\}$ at the time t implies that $E(x_i) = D(x_i)$, indicating alignment between the encryption and deletion processes at these points. Therefore, the polynomial validation confirms that the deletion criteria are satisfied.

The key decay function $f(t)$, defined as $e^{-\lambda t}$ with λ as the decay constant, influences $E(x)$ over time. The transformed polynomial $E'(x)$, affected by $f(t)$, models the degradation of the encryption key:

$$E'(x) = E(x) \cdot f(t).$$

Consequently, the differential polynomial $R'(x) = E'(x) - D(x)$ reflects the decaying effectiveness of $E(x)$ over time. As $t \to \infty$, $E'(x)$ converges to zero:

$$\lim_{t \to \infty} E'(x) = 0.$$

This convergence guarantees that, ultimately, $E(x)$ decays to an irrecoverable state, securing the deletion of data. The controlled decay mechanism embedded within $E(x)$ enforces data ephemerality, as $E(x)$ irreversibly intersects with $D(x)$, ensuring secure deletion within the system.

The deletion phase is mathematically verified by checking the integral condition:

$$\int_0^\infty E(x)\,dx = D(x),$$

over a relevant finite field $\mathbb{F}_p$, assuming the existence of points $x_i \in \mathbb{F}_p$ such that the condition holds:

$$\exists x_i \in \mathbb{F}_p.$$

This represents a sufficient condition for deletion validation under the conjunctive framework, ensuring that no unauthorized reconstruction of $E(x)$ is possible once decay takes effect.

The computational complexity of finding the roots of $R(x)$ lies at $O(N \log N)$, where N denotes the number of data points evaluated. This complexity ensures efficiency and the probability of incorrectly confirming deletion without genuine intersections is negligible (approximately of $\approx \frac{1}{|\mathbb{F}_q|}$), thus confirming the robustness of the conjunctive deletion strategy. □

4.4.3. FORMAL ANALYSIS

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 4.4.1 (Correctness)**.** *PBCS is correct (Def. 8).*

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 [70], 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 §4.4.1). Furthermore, only authorized COs can validate and execute deletion requests, preserving correctness under adversarial conditions, as proven by lemmas 1 to 12. □

Theorem 4.4.2 (Privacy of Inputs)**.** *PBCS is Λ-secure (Def. 9).*

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 [133] from the CP (See Theorems 1, 4, 6, 8, and 12). The CP can't distinguish the simulation from the real-world run

$$\Pr[V_{\text{real}} \neq V_{\text{sim}}] \leq \epsilon, \text{ where } \epsilon \text{ is negligible.}$$

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 intents 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 §13). The probability of success for malicious co-owners is negligible, as SSSS guarantees [70] 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).

□

4.4.4. 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 resolving 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, ensuring 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) by analyzing timing information, such as response times.
Winning Condition: $\mathcal{A}$'s probability of correctly guessing the operation type is bounded:

$$\mathbb{P}[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \epsilon,$$

where ϵ is negligible.

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 [133], while collaborative threshold enforcement ensures key recovery requires θ shares, with success probability for fewer shares bounded by

$$\mathbb{P}[\mathcal{A} \text{ reconstructs } K] \leq \frac{1}{|\mathbb{F}_q|}, (\textit{Theorem } 7).$$

4.5. RESULTS AND INSIGHTS

A prototype was developed to evaluate its efficacy in secure, conjunctive audience-specific data management and GDPR compliance. The prototype utilized AES-256-GCM for encryption and RSA-OAEP with a 2048-bit key and SHA-256, following NIST standards [71, 140–143]. 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*) IAM-Based Policies, (*ii*) Evaluation Metrics, and (*iii*) Experimental Results.

4.5.1. IAM-BASED POLICIES FOR PBCS PROTOCOL

IAM POLICY FOR UPLOAD

This IAM policy restricts upload permissions to the data owner, ensuring only authorized users can upload the encrypted data object to the provider. It enforces access based on request tags, allowing only requests with the correct role and policy tags.

Listing 4.1: IAM Policy for Upload

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Role": "DataOwner",
          "aws:RequestTag/Policy": "UploadPolicy"
        }
      }
    }
  ]
}
```

IAM POLICY FOR PREPROCESSING

The Cloud Provider (CP) initializes the preprocessing step to generate access control based on group signatures and IDs. It also generates the deletion polynomial to validate votes for deletion.

Listing 4.2: IAM Policy for Preprocessing

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Role": "CloudProvider",
          "aws:RequestTag/Task": "Preprocessing",
          "aws:RequestTag/ActionType": [
            "GenerateAccessPolynomial",
            "GenerateDeletionPolynomial"
          ]
        }
      }
    }
  ]
}
```

IAM POLICY FOR ACCESS CONTROL

Access control is designed to validate access requests. The validation uses group identifiers and cryptographic signatures to ensure that only authorized users can access the data.

Listing 4.3: IAM Policy for Access Control

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "NumericGreaterThanEquals": {
          "aws:RequestTag/VotesApproved": "2"
        },
        "StringEquals": {
          "aws:RequestTag/ActionType": "AccessControl",
          "aws:RequestTag/Policy": "ValidateAccessRequest",
          "aws:RequestTag/IdentifierType": "GroupSignature",
          "aws:RequestTag/AccessType": "Authorized"
        }
      }
    }
  ]
}
```

IAM POLICY FOR DOWNLOAD

Download requests are validated based on valid access approval from the access polynomial. Only authorized users can download the data.

Listing 4.4: IAM Policy for Download

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Role": "CoOwner",
          "aws:RequestTag/GroupIdentifier": "12345",
          "aws:RequestTag/AccessGranted": "true",
          "aws:RequestTag/Policy": "ValidAccess"
        },
        "NumericGreaterThanEquals": {
          "aws:RequestTag/VotesApproved": "2"
        }
      }
    }
  ]
}
```

IAM POLICY FOR DELETION

Deletion requires valid access and can be either majority-based or veto-based. Majority deletion requires at least threshold votes, while veto allows a single co-owner to delete the object.

Listing 4.5: IAM Policy for Deletion

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "NumericGreaterThanEquals": {
          "aws:RequestTag/VotesApproved": "2"
        },
        "StringEqualsIfExists": {
          "aws:RequestTag/Policy": "Deletion",
          "aws:RequestTag/DeletionType": "Majority",
          "aws:RequestTag/Role": "Co-Owner"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Policy": "Deletion",
          "aws:RequestTag/DeletionType": "Veto",
          "aws:RequestTag/Role": [
            "Co-Owner",
            "Admin"
          ],
          "aws:RequestTag/VotesApproved": "1"
        }
      }
    }
  ]
}
```

IAM POLICY FOR DELETION VALIDATION

The Co-Owners verify the final deletion status by checking the polynomial intersection. If the deletion is confirmed, the validation can be retrieved.

Listing 4.6: IAM Policy for Deletion Validation

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "NumericGreaterThanEquals": {
          "aws:RequestTag/VotesApproved": "2"
        },
```

```
        "StringEqualsIfExists": {
          "aws:RequestTag/Policy": "Deletion",
          "aws:RequestTag/DeletionType": "Majority",
          "aws:RequestTag/Role": "Co-Owner"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:DeleteObject",
      "Resource": "arn:aws:s3:::mybucket/datafile123",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Policy": "Deletion",
          "aws:RequestTag/DeletionType": "Veto",
          "aws:RequestTag/Role": [
            "Co-Owner",
            "Admin"
          ],
          "aws:RequestTag/VotesApproved": "1"
        }
      }
    }
  ]
}
```


4.5.2. 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 4.2). Parameters were empirically selected to achieve data inaccessibility over periods 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 [144] 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 $r(x) = P(x) \bmod G(x)$. 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 [145].

4.5.3. EXPERIMENTAL RESULTS

Our experiments targeted two areas: (*i*) Decay Analysis and (*ii*) Performance Evaluation.

Table 4.2: Parameter Space for Conjunctive Key Decay Used in Study.

Parameter	**Low Decay**	**Moderate Decay**	**High Decay**	**Exponential Decay**	**Sigmoidal Decay**
$\lambda(\text{hr}^{-1})$	[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
$\Delta\lambda(\text{hr}^{-1})$	+0.01%	+0.33%	+20%	+0.21%	+1.67%
$K(t)$ (Fig 4.6)	Long-term access	Medium-term access	Short-term access	$K_0 \cdot e^{-\lambda t}$	$\frac{K_0}{1+e^{-\lambda(t-T_{\text{mid}})/s}}$
$\mu_{50}(\text{min}^{-1})$	≤ 20	≤ 10	≤ 5	Sensitive to initial value	Slow start, rapid midpoint increase
$\mu_{100}(\text{min}^{-1})$	[80, 100]	[45, 60]	[10, 20]	Stabilizes after initial drop	Stable after midpoint $s \in [0.1, 10]$
$\mu_{200}(\text{min}^{-1})$	[140, 160]	[100, 115]	[30, 50]	High stability across time	Stable after midpoint
θ	0.10 (Longer access)	0.05 (Moderate access)	0.01 (Shorter access)	$[0.02, 0.05]$	$[0.03, 0.05]$
α_i	1.0 (Standard sensitivity)	1.5 (Moderate sensitivity)	2.0 (High sensitivity)	Adjusts with policy	Adjusts with policy
w_i	1.0 (Standard importance)	1.5 (Moderate importance)	2.0 (High importance)	Adjusts with policy	Adjusts with policy

Legend: 1) λ: Key decay. 2) s: Sigmoidal steepness. 3) K_0: Initial key. 4) T: Validity. 5) $\Delta\lambda$: Decay rate increase. 6) $K(t)$: Key effectiveness. 7) μ: Value stability. 8) θ: Threshold. 9) α_i: Sensitivity factor. 10) w_i: Share importance.

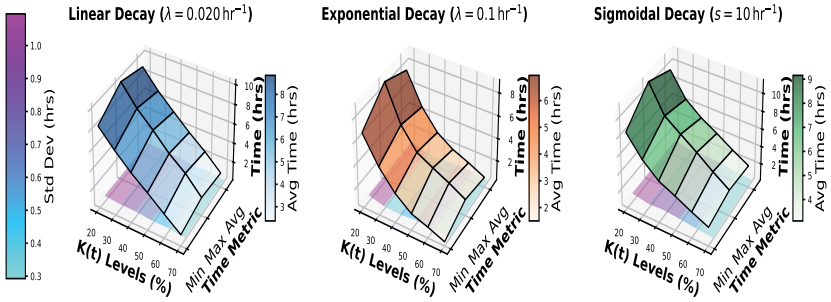


Figure 4.6: Run-time Statistics for Various Decay Factors Influencing Key Effectiveness ($K(t) \in [70\%, 20\%]$).

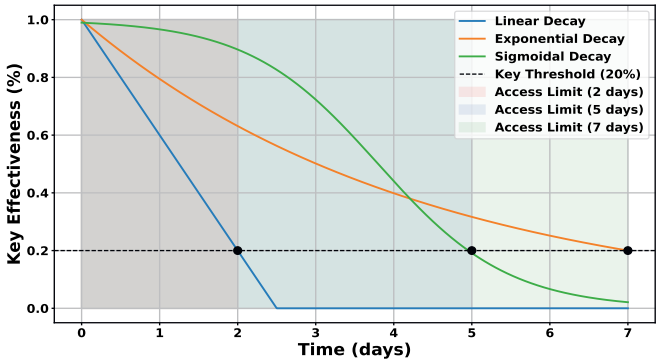


Figure 4.7: Key Effectiveness Over Time with Decay Rates, Highlighting Conjunctive Access Limits in 2 to 7 Days.

DECAY ANALYSIS

This analysis investigates how key ephemerality impacts data access control within the conjunctive framework of PBCS, essential for maintaining security and compliance in collaborative cloud environments. By evaluating 100 groups and 1000 co-owners, we explored distinct decay models tailored to address varied access and expiration scenarios.

a) *Key Effectiveness Over Time (Fig. 4.6):* 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, ideal for swift data obsolescence. Sigmoidal decay showed an *S*-shaped curve, with initial resistance followed by rapid decline, supporting adaptable security protocols. The runtime analysis (minimum, maximum, average,

Table 4.3: 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

and deviations) demonstrates consistent performance, with all decay models maintaining stability as effectiveness decreases. Detailed numerical results and comparative metrics are presented in Table 4.3.

b) *Group Dynamics and Key Ephemerality (Fig. 4.7):* 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 two days, making it ideal for time-sensitive tasks requiring 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 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 (Fig. 4.8):* 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

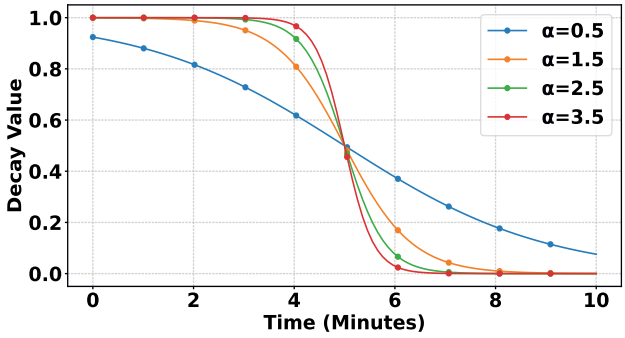


Figure 4.8: Impact of Sigmoidal Decay on Key Sensitivity Across Different Sensitivity Levels (α_i) Over Time.

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.

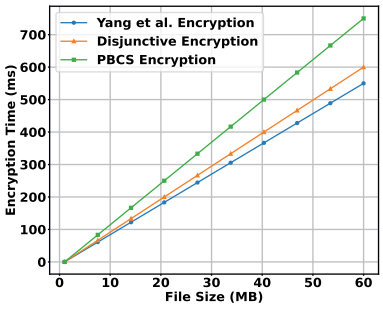
PERFORMANCE EVALUATION

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

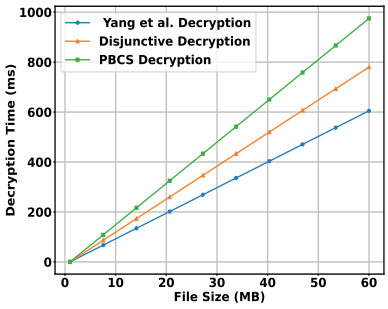
Computational Time (Fig. 4.9). This assessment focused on the operational time through the phases, including encryption, decryption, deletion, and validation.

a) *Encryption Time (Fig. 4.9(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.

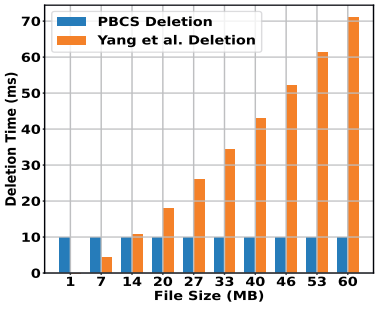
b) *Decryption Time (Fig. 4.9(b)):* The decryption times mirror the encryption pattern, with PBCS again incurring the highest decryption cost due to the need for complex access control verifications and policy-based checks ($O(n^2)$). At 60 MB, decryption takes approximately 975 ms, mainly due to the polynomial complexity of reconstructing keys



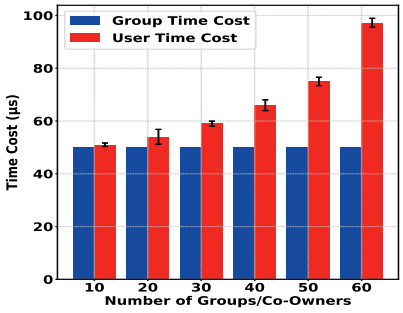
(a) Encryption Comparison.



(b) Decryption Comparison.



(c) Deletion Comparison.



(d) Validation Comparison.

Figure 4.9: Performance Evaluation of Various Time Costs Across Different Schemes and Validations in PBCS.

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.

c) *Deletion Time (Fig. 4.9(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. operate with $O(n \log n)$ complexity due to the Merkle tree structure required for deletion, with time increasing logarithmically with file size (e.g., 70 ms for 60 MB). Each deletion requires recalculating hashes along the Merkle tree,

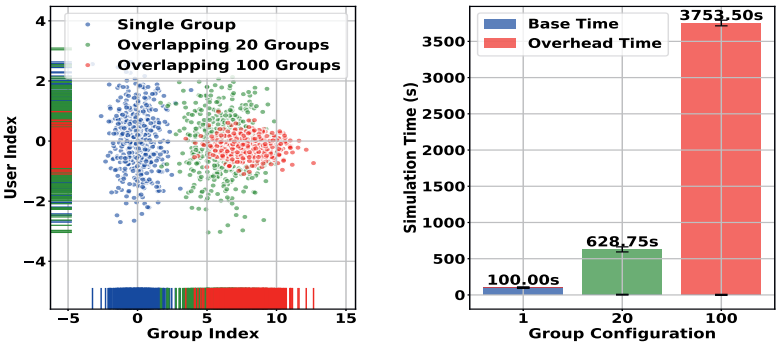


Figure 4.10: Scalability in Complex Configurations: Impact of Overlapping Co-Owners Across Various Groups.

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 (Fig. 4.9(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 co-owner numbers grow, the time cost rises with the Lagrange polynomial's increasing degree, adding computational demands per user share. For $n = 5$ users, time cost is around $55\,\mu$s; for $n = 60$, it rises to $100\,\mu$s. This increase is due to the linear overhead in deletion validation as more co-owners are included.

Scalability (Fig. 4.10). We tested PBCS under complex configurations with overlapping co-owners across multiple group structures, totaling 1000 trials.

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, 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, highlighting areas where computational overhead 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. Future work will explore parallel processing, batch validations, and reducing redundancy in key reconstruction.

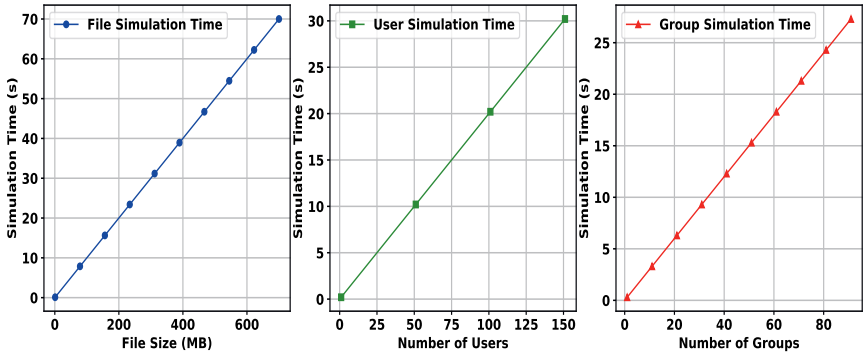
d) *Linear Scalability:* In the scalability assessment of the conjunctive scheme, we evaluated three pivotal dimensions to reflect the scheme's adaptability to increasing operational demands (See Figure 4.11(a)). *a) Scale of File per Group:* Empirical data reveals a linear increase in operational time 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).

e) *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 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 4.11(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.

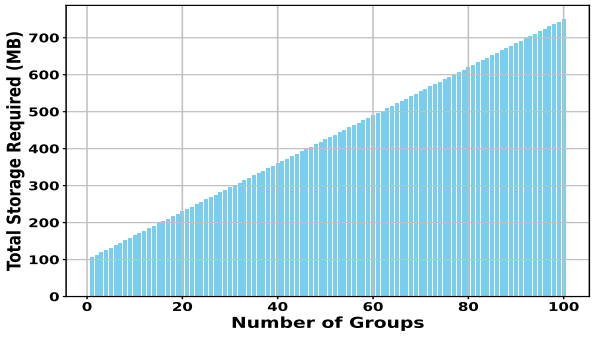
4.6. RELATED WORK

4.6.1. DATA DELETION MECHANISMS.

Yang et al. [146] proposed a blockchain-based scheme enabling publicly verifiable data deletion in untrustworthy cloud environments. Xiong et al. [83] tailored key derivation encryption for IoT flash memory hierarchies, enabling key deletion upon data expiration. Yu et al. [147] used attribute-based encryption (ABE) to ensure data deletion via



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



(b) Total Storage Across Groups.

Figure 4.11: Linear Scalability Trends and Total Storage Used.

attribute revocation, while Yang et al. [118] utilized counting Bloom filters for secure deletion. Kaushik and Gandhi [148] designed a capability-based system with assured deletion through user revocation. Ma et al. [149] introduced secure data deletion with verification (SDVC) using an attribute association tree for rapid revocation and re-encryption. Such and Criado [123] employed conflict resolution strategies based on shared data sensitivity. Enrico et al. [150] suggested fragment slicing with iterative re-encryption for efficient revocation. Filipe [151] developed Scramble, enabling social network users to encrypt, revoke, and delete data. Olteanu et al. [152] 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.

4.6.2. COLLABORATIVE ACCESS.

Huang et al. [153] 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. [134] proposed a CP-ABE scheme for public cloud storage, allowing users to combine attributes for collaborative access control. Carminati et al. [122] introduced collaborative security policies in online social networks, involving multiple users in access decisions based on relationships. Ilia et al. [121] presented SAMPAC as a privacy-preserving data-sharing system where co-owners set access controls collaboratively, managed by a TTP (i.e., KMS). In He et al. [154], 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 4.4: Comparison with Previous Works [87] & [82].

Feature	**Disjunctive [87]**	**Yang et al. [82]**	PBCS
Model	Amortized	Amortized	Amortized
TTP	✗	✗	✗
Verifiability [155]	✗	$\mathcal{P}_{\sqcap}$	$\mathcal{P}_{\triangledown}$
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 & Deletion Policy	✗	✗	✓
Autonomy	Low	Low	High

1) n: Number of data blocks. **2)** $\mathcal{P}_{\triangledown}$: Private verifiability. **3)** $\mathcal{P}_{\sqcap}$: Public verifiability.

4.6.3. COMPARATIVE ANALYSIS OF SCHEMES.

We compare the Disjunctive Scheme [87], Yang et al. [82], and PBCS across key metrics, as shown in Table 4.4. All schemes adopt an amortized model, avoiding TTP reliance. PBCS excels in private verifiability, dynamic membership management, and collaborative deletion, ensuring co-owner autonomy and flexible access control. It is particularly suited for scenarios demanding stringent policy enforcement and audience-based data forgetting. 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.

4.7. CONCLUSION AND FUTURE WORK

In response to the need for advanced data outsourcing in collaborative settings like literary collectives, this paper 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.

5

VERIFIABLE CO-OWNED DATA DELETION IN MULTI-CLOUD ENVIRONMENTS

The increasing adoption of multi-cloud environments and the rise of collaborative data ownership introduce critical challenges in the verifiable deletion of shared data. Existing approaches predominantly address individual ownership and often rely on simplistic one-bit result protocols where a deletion command merely outputs success or failure, turning the deletion into a black box without proper verification. This chapter tackles the problem of secure processing and verifiable deletion of shared outsourced data in multi-cloud systems. We design a framework that enables a data owner to outsource encrypted data to multiple co-owners, who perform computations directly within their respective cloud providers—ensuring that sensitive data never leaves the cloud. Our system leverages readily available cloud Hardware Security Modules *(HSMs) to manage cryptographic keys from generation to controlled destruction—ensuring data remains inaccessible beyond its intended use. Secure Enclaves execute on-cloud data computation, preventing local copies and unauthorized exposure. Encrypted data is structured within a fixed storage model, ensuring controlled allocation and strict storage constraints. When data expires or must be deleted to meet regulatory conditions, our framework initiates zero-residuals*

This chapter is based on *two papers:* "Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment" by Darwish, M.A., Markatou, E.A., & Smaragdakis, G. in Proceedings of the 18th European Workshop on System Security 2025 [156] and "From Creation to Deletion: Secure and Verifiable Data Management in Cloud Environments" by Darwish, M.A., Markatou, E.A., & Smaragdakis, G., which is under review.

permuted overwriting to remove the data traces irreversibly. Verifiability is achieved at two levels: locally, Bounded Merkle Hash Tree (BMHT) ensures bounded storage and verifiable deletion within each cloud provider. Globally, Global Merkle Forest (GMF) aggregates BMHT roots across providers, enabling consistent verification. The data owner maintains a comprehensive log of BMHT root hashes, enabling independent verification of secure deletion across the multi-cloud environment. Finally, we present a formal analysis, an experimental evaluation, and a cost assessment to validate the effectiveness of our scheme.

5.1. INTRODUCTION

Modern technologies and services rely on data-intensive systems powered by algorithms that process massive volumes of information. Digital data is now critical to many organizations and services, with far-reaching implications for our daily lives [157]. However, as more sensitive and personal data is sent to the cloud, worries about data integrity, availability, and secure deletion have increased substantially [158]. Among these problems, securely and provably erasing data is critical for protecting users' data privacy during the end of its relevance [105, 159–161]. Failure to ensure provable data deletion can have serious consequences, including privacy breaches, security risks, and regulatory violations [162–164]. The difficulty of secure and provable data erasure has led researchers to suggest numerous solutions. Many of these approaches aim to ensure compliance with GDPR [165, 166]. These solutions can be broadly classified as the following approaches:

a) Deletion by Unlinking [167–169]: This method removes the link between the data and its underlying file system. When a user issues a deletion command, the system responds with a one-bit result, either success or failure. However, while the data is logically removed, it remains physically intact in the storage medium, leaving it vulnerable to recovery through forensic tools. This approach fails to provide reliable assurance of secure deletion.

b) Deletion by Cryptography [53, 56, 87, 170, 171]: Data is first encrypted, and its security depends on the encryption key. The data becomes unrecoverable once the key is destroyed, though the ciphertext remains stored in the cloud. This approach is beneficial when duplicate copies of data exist across distributed locations, making it impractical to overwrite every copy. Cryptography shifts the challenge from deleting large amounts of data to deleting a short encryption key. However, the key must be securely deleted, which remains a critical challenge.

c) Deletion by Overwriting [172–177]: In this approach, old data

is replaced with new or random data of the same size to ensure it is irretrievable. Unlike cryptographic deletion, which makes data inaccessible, overwriting physically removes it. However, improper implementation may leave residual traces (i.e., incomplete removal of data or unintended storage expansion) of the original data. Advanced microscopic tools can exploit these traces, exposing the physical remnants and compromising the data's integrity [53].

These limitations become even more pronounced in multi-cloud systems, where co-owned data—shared among multiple stakeholders across diverse providers—encounters significant challenges [178–180]. Prior studies lack synchronized management to address inconsistent deletion practices, cross-provider protocol gaps, and the need for verifiable proofs at each step [181]. For example, European hospitals collaborate with research institutes and universities on health data research to: **(i)** identify anomalies and outliers, **(ii)** enhance privacy protections, and **(iii)** inform public health policy. The data, co-owned by multiple stakeholders, is stored across different providers, such as AWS [182], Azure [183], or GCP [184], depending on operational needs or resources provided. Europe enforces strict data protection laws, starting with GDPR and recently *European Health Data Space* (EHDS) [38, 185], which mandate secure data sharing and retention. Under EHDS, hospitals (data owners) must request deletion from all relevant cloud providers when data expires, ensuring compliance and transparency. Providers verify consent, execute secure deletion, and return cryptographic proofs of data deletion and storage integrity. Co-owners and regulators use these proofs to ensure adherence to EHDS policies. Recent real-world incidents further emphasize the need for verifiable deletion mechanisms. 23andMe, a personal genomics and biotechnology company that had genomic information of more than 14 million people [186], filed for bankruptcy filing (March 2025); BBC reported that users raced to delete their sensitive genetic data but faced technical barriers and lingering doubts about whether deletion had actually occurred [187].

Such scenarios highlight the need for data-sharing mechanisms that enable: **(1)** tamper-proof data sharing, **(2)** secure processing, **(3)** consistent and auditable updates, and **(4)** verifiable deletion when data is no longer needed, in line with data minimization principles. *We propose a comprehensive data-sharing framework to meet these requirements.*

High-Level Idea. We propose a comprehensive framework for secure and provable co-owned data deletion in multi-cloud environments, ensuring that data never leaves the cloud and computations stay within each respective provider's infrastructure. The system supports a Data Owner (DO) and multiple Co-Owners (COs), each associated with a distinct provider with separate infrastructures. No local copies are ever

created, and providers do not know the actual data or computations. To achieve this, we integrate *Hardware Security Modules (HSMs)* [188] at each CP to securely handle cryptographic keys from generation to controlled destruction, ensuring data remains inaccessible after expiry.

Secure Enclaves [189] enable on-cloud computations without external data storage, but are costly; we limit their use to critical operations. For instance, AWS Enclaves cost $0.18/hr in our setup, whereas a standard EC2 instance costs only $0.0084/hr, making enclaves significantly more expensive. In order to ensure zero residual overwriting, we follow the guidelines of NIST Special Publication (SP) Computer Science Series 800-88 "Guidelines for Media Sanitization" [190].

Our framework enforces a structured 3-cycle process: **(i)** pseudorandom function (PRF) masking to randomize data patterns, **(ii)** bit-order reversal to disrupt identifiable structures, and **(iii)** pseudorandom permutation (PRP)-based repositioning to ensure destruction of residual information. While our scheme enforces bounded storage, filesystem- or firmware-level remapping (e.g., SSD FTLs, EBS) may allocate new physical blocks, preventing guaranteed physical overwriting. Thus, we adopt a logical sanitization model consistent with NIST SP 800-88 Rev.2: all data remains encrypted at rest, and cryptographic erasure (CE) is enforced via HSM key destruction. This dual-layer strategy supports hybrid storage (e.g., SSDs, HDDs), provides verifiable proof of sanitization beyond opaque CE, and ensures stale blocks are actively overwritten. It aligns with cloud standards, supports data minimization, and mitigates "harvest-now, decrypt-later" risks [24, 172]. Furthermore, to verify the 3-cycle process, each overwriting step is correctly executed, and a new hash root is recalculated. Each CP constructs a *Bounded Merkle Hash Tree (BMHT)*, a structured variant of *Merkle Hash Tree* (MHT) [191], to limit storage expansion. Rather than enforcing real-time authorization, the system maintains verifiable logs of updates and deletions, allowing owners to detect discrepancies later. Any copies outside the protocol remain unusable without the private keys securely stored in HSMs. Verifiable deletion is achieved locally via BMHT at each CP and globally via *Global Merkle Forest (GMF)*, which aggregates BMHT roots for consistency. The system generates proofs for storage, membership/non-membership, overwriting, and global integrity.
In summary, our paper makes the following key contributions:

- We present a framework for provable co-owned data deletion, ensuring co-owned data never leaves the cloud via HSMs for key management and Secure Enclaves for on-cloud processing.
- We introduce BMHT to enforce strict storage constraints and provide cryptographic proofs for storage integrity, membership, non-membership, and overwriting, ensuring verifiable deletion at the provider level.

- We design a zero-residuals overwriting mechanism that guarantees irreversible data erasure, preventing recovery.
- We develop GMF to aggregate deletion proofs across providers, ensuring global verifiability and consistency in multi-cloud environments.
- We implement and evaluate the framework in a cloud environment, assessing its scalability, computational efficiency, proof generation overhead, and cost-effectiveness.

5.2. DESIGN GOALS

This study proposes a provable deletion that ensures zero-residual overwriting without provider trust by leveraging hardware-based key, trusted execution, and multi-layer proof verification, assuming the following settings:

5.2.1. ASYNCHRONOUS AND SECURE COMMUNICATION

The solution supports asynchronous environments, handling communication delays among cloud providers and stakeholders. All proof exchanges, key management, and data transmissions are secured using Transport Layer Security (TLS) or equivalent protocols [192].

5.2.2. THREAT MODEL

Byzantine Cloud Providers (CPs): CPs may act as Byzantine adversaries by deviating from protocols, retaining data, falsifying proofs, or aborting computations to disrupt availability. Our framework remains resilient even if CPs collude, as it does not rely on provider trust. Instead, encryption keys are uniquely managed per provider, ensuring that raw data remains inaccessible. CPs store only encrypted data and execute computations within Secure Enclaves, preventing unauthorized access. While the system cannot prevent aborted operations by a CP, it mitigates them via encrypted data migration to a new provider, ensuring liveness with minimal setup overhead.

Trusted Data Owners (DOs): Data owners are considered trusted entities responsible for initiating data sharing, storage, and deletion processes. They coordinate data management activities with co-owners and cloud providers.

Trusted-But-Forgetful Co-Owners (COs): COs, such as research institutions, are assumed to comply with the framework's execution model—processing data exclusively inside the Secure Enclaves without extracting local copies. Each CO has a key pair (PKI) that ensures authenticated and encrypted communication with the DO. This reflects collaborative settings where partners are operationally trusted but may

forget to delete temporary data. *To strengthen this model, enclave attestation ensures that only DO-approved code can access data. While we outline a mitigation strategy in §5.6, formal guarantees against fully malicious COs remain future work.*

Trusted Hardware Manufacturers: Trusted Hardware Manufacturers provide HSMs for secure key management and Secure Enclaves for isolated execution, ensuring secure communication and computations. While CPs provide access to this hardware, the framework assumes HSMs operate as NIST Federal Information Processing Standards (FIPS) 140−2 Level 3 certified [193], guaranteeing key isolation and irreversible destruction. It also assumes Secure Enclaves enforce in-memory data processing, preventing persistence or external access, with execution isolated from CPs and the host OS. These security assumptions rely on the certified security of the manufactured hardware, ensuring CPs cannot extract encryption keys or decrypted data, even if they interact with the hardware.

5.2.3. SECURITY MODEL

We focus on cryptographic proofs for our security model, assuming trusted HSMs and Secure Enclaves.

Correctness: Correctness holds when a valid proof, generated by the prover (CPs), enables the verifier (DOs or COs) to validate storage, membership, non-membership, and overwriting operations. Let $D = \{d_1, d_2, \ldots, d_n\} \in \mathbb{B}^*$ be the dataset stored in the system, where $d_i = (k_i, D_i)$ is a unique key-value pair stored as a leaf in the BMHT. For a leaf at index i, the hash is: $H_{\text{leaf}} = \text{Hash}(k_i \parallel D_i)$. Let $\mathcal{H}_{\text{root}}$ denote the BMHT root hash. Correctness guarantees the following for all proofs:

1. Storage Proof ($\pi_{storage}$): If the BMHT constructed on dataset D has root $\mathcal{H}_{\text{root}}$, then the verifier checks:

$$V_{\pi_{\text{storage}}}(D, \mathcal{H}_{\text{root}}, \pi_{\text{storage}}) = 1.$$

where $V_{\pi_{\text{storage}}}$ is the verification function, and π_{storage} is the proof that $\mathcal{H}_{\text{root}}$ correctly encodes all elements of D.

2. Membership Proof ($\pi_{membership}$): Confirms that a key-value pair $d_i = (k_i, D_i)$ is present in D:

$$V_{\text{membership}}(d_i, \mathcal{H}_{\text{root}}, \pi_{\text{membership}}) = 1.$$

3. Non-Membership Proof ($\pi_{non\text{-}membership}$): Confirms that a queried key $k \notin D$, using its predecessor k_{pred} and successor k_{succ}:

$$V_{\text{non-membership}}(k, \mathcal{H}_{\text{root}}, \pi_{\text{non-membership}}) = \begin{cases} 1, & k_{\text{pred}} < k < k_{\text{succ}}, \\ 0, & \text{otherwise.} \end{cases}$$

4. Overwriting Proof ($\pi_{overwrite}$): Validates that an old key-value pair $(k_{\text{old}}, D_{\text{old}})$ has been securely replaced by a new pair $(k_{\text{new}}, D_{\text{new}})$:

$$V_{\text{overwrite}}((k_{\text{old}}, D_{\text{old}}), (k_{\text{new}}, D_{\text{new}}), \mathcal{H}_{\text{root}}, \pi_{\text{overwrite}}) = 1.$$

Security: Security ensures that no malicious cloud provider can forge proofs, tamper with BMHT or GMF, access stored data, or observe computations. It guarantees verifiable proofs and ensures that overwritten data remains indistinguishable from random noise.
Let π denote a valid proof, which can be of one of the four types:

$$\pi \in \{\pi_{\text{storage}}, \pi_{\text{membership}}, \pi_{\text{non-membership}}, \pi_{\text{overwrite}}\}.$$

Representing storage verification, membership confirmation, non-membership validation, or overwriting proof, respectively.
Global Integrity Enforcement: The Global Merkle Forest (GMF) aggregates the root hashes from all cloud providers, ensuring verifiability and consistency across storage environments:

$$\mathcal{H}_{\text{GMF}} = \text{Hash}(\mathcal{H}_{\text{root}}^{(1)} \| \cdots \| \mathcal{H}_{\text{root}}^{(M)}),$$

where each $\mathcal{H}_{\text{root}}^{(i)}$ represents the root hash of the BMHT maintained by the i-th provider. This aggregation guarantees that data modification, omission, or forgery can be detected across M providers.
Given the integrity constraints imposed by GMF, the probability of an adversarial cloud provider generating a false proof that passes verification is negligibly small. Specifically, for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ attempting to produce a forged proof π':

$$\Pr\left[V_{\text{proof}}(\pi') = 1 \mid \pi' \neq \pi \text{ or } \mathcal{H}_{\text{root}}^{(i)} \notin \mathcal{H}_{\text{GMF}}\right] \leq \epsilon(k),$$

where $\epsilon(k)$ is a negligible probability, meaning that as the security parameter k increases, the probability of a successful forgery or integrity violation asymptotically approaches zero.
Overwriting is cryptographically verifiable through BMHT proof updates, ensuring that new data replaces old data. However, zero-residual overwriting is a system-level measure that follows NIST. The security of overwriting instead follows PRF and PRP security assumptions:

$$\Pr\left[\mathcal{A} \text{ inverts } F\right] \leq \epsilon(k).$$

5.3. PRELIMINARIES

5.3.1. BOUNDED MERKLE HASH TREES (BMHT)

The BMHT extends the traditional MHT [191] by incorporating a fixed and sorted tree structure, ensuring efficient storage management without

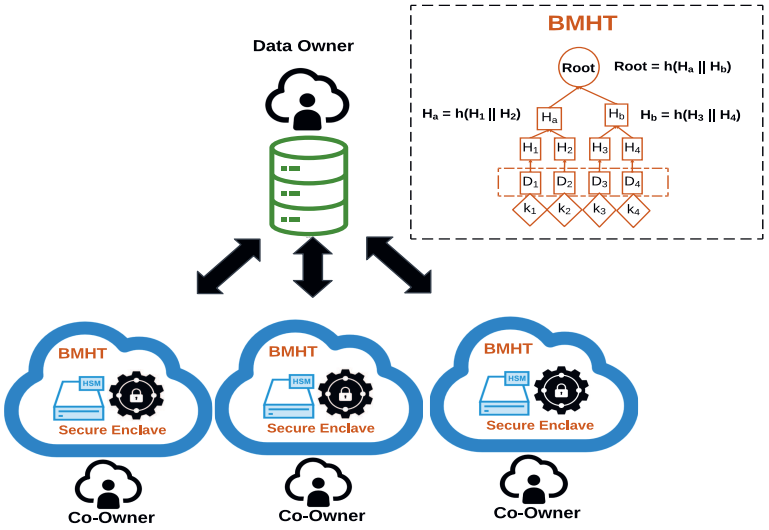


Figure 5.1: The Framework Employs BMHT for Fixed Storage, HSMs for Secure Key Management, and Secure Enclaves to Ensure On-Cloud Data Access Without Local Copies.

concerns of dynamic growth. The BMHT consists of a constant number of leaf nodes, N, each representing a predefined storage block of size S, ensuring a predictable and enforceable tree height by the owners. The leaf nodes are sorted in ascending order by their keys to facilitate efficient proofs. With N leaf nodes, the total storage capacity is given by: $C = N \cdot S$. This fixed structure guarantees consistent storage limits and avoids the growth issues associated with traditional MHTs in constrained environments. Each leaf node hash is computed as:

$$H_{\text{leaf}_i} = \text{Hash}(k_i \parallel D_i),$$

where k_i is the binary key and D_i is the data block (which will be encrypted). Including the key in the hash ensures positional integrity. Internal nodes derive their hashes recursively from their child nodes, culminating in a root hash H_{root} that serves as a cryptographic commitment to the integrity of the entire dataset. The key differences between the BMHT and the traditional MHT are: **(i)** BMHT maintains a fixed number of leaf nodes, enforcing strict storage limits, **(ii)** BMHT's sorted leaf structure enables efficient non-membership proofs via predecessor-successor indexes verification, and **(iii)** The fixed order ensures smaller and more predictable proof sizes for non-membership and overwriting operations compared to a dynamic MHT.

5.3.2. GLOBAL MERKLE FOREST (GMF)

The GMF provides a global proof mechanism by aggregating the local root hashes of BMHT instances from multiple cloud providers, ensuring tamper-resistant evidence of data operations across a multi-cloud environment. Assume there are M providers, where the j-th provider manages a BMHT with root hash $H_{\text{root}}^{(j)}$. The global root hash H_{GMR} is then computed as:

$$H_{\text{GMR}} = h\left(H_{\text{root}}^{(1)} \parallel H_{\text{root}}^{(2)} \parallel \cdots \parallel H_{\text{root}}^{(M)}\right),$$

where $h(\cdot)$ denotes a cryptographic hash function and $\parallel$ represents concatenation. At any time, only one H_{GMR} exists, stored locally for verification and in HSMs for integrity protection. This structure integrates the fixed storage enforcement at each provider via the BMHT while offering a unified cryptographic commitment to the global data state. To verify the integrity of a data block D_i stored at provider j, the verification proof includes the path from the leaf node L_i (representing D_i) up to the local BMHT root $H_{\text{root}}^{(j)}$, as well as the aggregated hashes required to reconstruct H_{GMR}. The proof is considered valid if:

$$H_{\text{GMR}} = h\left(\pi_{\text{global}}\right),$$

where π_{global} combines the local proofs from the BMHTs of all providers. This approach guarantees transparent and verifiable operations (e.g., overwriting) across providers while preserving the independence of each BMHT.

5.4. SYSTEM ARCHITECTURE

This section presents the comprehensive architecture of the proposed solution. Figure 5.1 illustrates the system's key components and operational workflow.

Initialization: The DO initializes the system by dividing the dataset into fixed-size blocks (e.g., 256 MB) based on estimated project or data size and encrypts each block using a hardware-issued key. The encrypted blocks are mapped to a local Bounded Merkle Hash Tree (BMHT) leaf node, which enforces fixed storage. We apply the padding to ensure a symmetric tree structure and consistent block sizes. The DO computes the BMHT root and shares it with Co-Owners, allowing them to verify storage integrity. The DO retains key mappings and metadata for proof verification.

Preprocessing: The DO prepares the data by associating each block with a unique key and sorting the BMHT leaf nodes in ascending order by their keys. This facilitates efficient proof generation for both membership and non-membership verification.

Table 5.1: Notation Summary.

Notation	Description
D_i	i-th data block
D'_i	Padded data block ($\|D'_i\| = S$)
P_i	Padding for D_i
S	Fixed block size
C	Total capacity
N	Number of blocks
K	Symmetric encryption key
E_i	Encrypted block
k_i	Ordering key for E_i
τ	HSM-signed storage commitment
PK_j	Public key of co-owner j
SK_j	Private key of co-owner j
$f_k(x)$	Pseudorandom function (PRF)
$\pi_k(x)$	Pseudorandom permutation (PRP)
K_{session}	Session key
k	Security parameter
σ	Standard deviation in test
Δ_i	i-th segment of D_{new}
T	Test statistic function
$p_{i,j}$	p-value from test j on segment Δ_i
α	Significance threshold

Data Sharing: The DO securely shares data with COs using their public keys (PKI) for encryption via HSMs, ensuring data remains within the cloud until expiration. The HSMs decrypts data internally, without exposing it to the CP, and immediately re-encrypts it using a session-specific key before any processing internally (i.e., encryption/decryption occurs inside HSMs). The Secure Enclave is used only when data needs to be processed, enabling secure data processing while ensuring no local copies are created. The DO periodically publishes a signed digest of sorted key identifiers and BMHT root, allowing COs to verify CP-provided membership/ non-membership proofs.

Outsourcing: The CP computes the BMHT root hash over the encrypted blocks via HSMs to enable verifiable integrity checks by the DO and COs, who can request and verify proofs when needed. The BMHT structure enforces fixed storage, preventing uncontrolled expansion of stored data or an increase in the number of leaf nodes.

Data Analysis: Secure Enclaves facilitate confidential on-cloud data processing without exposing sensitive information or creating local copies. Instead of decrypting data outside the enclave, a session-specific symmetric key is generated by HSMs to encrypt data within the enclave before processing. This key ensures that the enclave can process encrypted data without revealing the PKI keys. The key is

securely destructed upon request, preventing unauthorized access and ensuring confidentiality.

Proofs Request: The CP provides BMHT roots periodically to be verified later when requested by the DO or COs, ensuring freshness and avoiding stale proofs. The CP does not proactively provide stored proofs but instead generates them dynamically upon request:

a) *Storage Proof:* Confirms that the data remains within the BMHT structure, ensuring fixed storage limits are intact.

b) *Membership Proof:* Verifies the inclusion of specific data blocks using their BMHT tree path.

c) *Non-Membership Proof:* Confirms specific data exclusion by verifying its predecessor and successor in the sorted BMHT. DOs or Co-Owners (COs) verify these proofs to ensure storage limits and data integrity.

Data Overwriting: Expired or updated data is securely erased using zero-residual overwriting, following the guidelines of NIST SP 800 — 88 [190]. A three-stage process ensures complete removal: first, PRF masking obscures data patterns; second, bit reversal disrupts residual structure; and third, entropic shuffling (i.e., randomly reordering data elements to maximize randomness) eliminates any remaining traces. The BMHT root is updated, with overwriting verified via membership (new data) and non-membership (old data) proofs. Overwriting can be triggered by expiration time or by uploading new data from DO.

Global Proof Synchronization: The GMF aggregates BMHT roots across CPs, ensuring consistent global storage verification. This guarantees that all parties remain consistent, even in asynchronous communication environments.

5.5. SCHEME DESCRIPTION

The framework ensures storage enforcement, secure execution, and verifiable deletion. It employs HSMs for secure key management, enforces fixed storage constraints, utilizes BMHT for proof generation, and leverages secure enclaves for controlled decryption. Zero-residual overwriting via PRFs and PRPs prevents recoverable traces. GMF synchronizes deletion proofs across providers for consistency. Figure 5.2 presents a complete system structure, and Table 5.1 summarizes key notations used.

5.5.1. INITIALIZATION

The initialization phase establishes fixed storage, encrypts data blocks, and constructs the Bounded Merkle Hash Tree (BMHT) for efficient proof generation and secure storage.

Initialization$(\lambda, N, S, D, \{PK_i, SK_{\mathbf{HSM}}\})$**:**

$\lambda \in \mathbb{N}$ (security parameter), $N, S \in \mathbb{N}$ (blocks, block size),
$C = N \cdot S$ (total storage capacity), $D_i \in \mathbb{B}^*$ (data blocks),
$k_i \in \mathbb{B}^*$ is the key index assigned to D_i,
PK_i, SK_i (public/private key pairs for co-owners),
SK_{HSM} (HSM private key for signing operations).

For each $i \in [N]$:

Pad: $D_i' = D_i \parallel \text{pad}(S - |D_i|)$, ensuring $|D_i'| = S$.

Encrypt: $E_i \leftarrow \text{Enc}(D_i', PK_i)$.

Compute leaf hash: $H(L_i) = h(k_i \parallel E_i)$.

BMHT Construction: $H_{\text{root}} \leftarrow \text{BMHT}(\{H(L_i)\}_{i=1}^{N})$.

Storage Commitment: $\tau \leftarrow \text{Sign}_{\text{HSM}}(H_{\text{root}} \parallel N \parallel S)$.

Output: $H_{\text{root}}, \tau, \{E_i\}_{i=1}^{N}, SK_{\text{HSM}}$.

Preprocessing$(N, S, \{E_i\})$**:**

Map blocks: $L_i \leftarrow E_i, \quad \forall i \in [N]$.

Sort by keys: Order $\{L_i\}_{i=1}^{N}$ in ascending order by k_i.

Output: $H_{\text{root}}, \{H(L_i)\}_{i=1}^{N}$.

Proof Generation$(H_{\text{root}}, \{H(L_i)\}, SK_{\mathbf{HSM}})$**:**

Storage proof π_{storage}: $H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau)$.

Membership proof: $\pi_{\text{mem}} = \{H_{\text{path}}\}$.

Non-membership proof: $\pi_{\text{non-mem}} = \{H_{\text{pred}}, H_{\text{succ}}\}$.

Zero-residual overwrite: $D_i \leftarrow \pi_k(\text{ReverseBits}(D_i \oplus \text{PRF}(k, H(D_i)))); \forall i$.

Overwrite proof: $\pi_{\text{overwrite}} = \pi_{\text{mem}} \cup \pi_{\text{non-mem}}$.

Output: $\pi = (\pi_{\text{storage}}, \pi_{\text{mem}}, \pi_{\text{non-mem}}, \pi_{\text{overwrite}})$.

Proof Verification$(\pi, \{H(L_i)\}, H_{\text{root}}, \{PK_i\})$**:**

Membership check: $H_{\text{path}}^{\text{comp}} \stackrel{?}{=} H_{\text{root}}$.

Non-membership check: $k_{\text{pred}} < k < k_{\text{succ}}$, $H_{\text{pred}}, H_{\text{succ}} \stackrel{?}{=} H_{\text{root}}$.

Overwrite check: $\pi_{\text{overwrite}} \stackrel{?}{=} \pi_{\text{mem}} \cup \pi_{\text{non-mem}}$.

Storage proof check: $H'_{\text{root}} \stackrel{?}{=} \text{Hash}(H_{\text{root}} \parallel \tau)$.

Decision: return `VALID` if all checks pass.

Figure 5.2: `Provable_Function` Overview for Secure Deletion in Multi-Cloud with Zero-Residuals & Verifiability.

Fixed Storage: Let the dataset be $D = \{d_1, d_2, \ldots, d_m\}$ in a dedicated storage medium (i.e., used by CP), where each D_i is a data block. The dataset is divided into fixed-sized storage blocks, each of size S. The total storage capacity is $C = N \cdot S$, where $N = \lceil C/S \rceil$ is the number of leaf nodes. Each block D_i is padded as much as necessary to ensure symmetry as follows:

$$D'_i = D_i \parallel P_i, \quad P_i = S - |D_i|, \quad |D'_i| = S.$$

Encryption: Each padded block D'_i is encrypted using a symmetric key K generated by the HSMs: $E_i = \text{Enc}_K(D'_i)$, where $\text{Enc}_K(\cdot)$ denotes the symmetric encryption function.

BMHT Construction: The BMHT consists of N fixed and sorted leaf nodes, each representing an encrypted block E_i with an assigned key k_i. The hash of each leaf node is: $H_{\text{leaf}_i} = \text{Hash}(k_i \parallel E_i)$. Sorting the leaf nodes by their keys k_i facilitates efficient proofs verifications (i.e., $\{H_{\text{leaf}_1}, H_{\text{leaf}_2}, \ldots, H_{\text{leaf}_N}\}$ where $k_1 < k_2 < \ldots < k_N$). Internal nodes are computed recursively as: $H_{\text{node}} = \text{Hash}(H_{\text{left}} \parallel H_{\text{right}})$, with the root hash serving as a commitment to data integrity: $H_{\text{root}} = \text{Hash}(H_{\text{left-subtree}} \parallel H_{\text{right-subtree}})$.

Tree Symmetry: To maintain a balanced tree, the height h is determined by: $h = \lceil \log_2(C/S) \rceil$. Symmetry ensures consistent proof sizes and predictable storage management.

Storage Extension: If additional capacity is required beyond C, a new storage instance with capacity $C' = N' \cdot S$ must be initialized, and all existing encrypted blocks $\{E_i\}$ are transferred to the new fixed-size layout via on-cloud computations. The original BMHT is discarded, and a new BMHT is constructed over the reallocated storage nodes.

5.5.2. VERIFICATION VIA BMHT AND PROOFS

After data is stored, in addition to the DO, the CP also constructs the BMHT to support storage, membership, and non-membership proofs, while the DO provides COs with a signed, sorted list of key identifiers to verify that the CP's proofs reference the correct keys.

Storage Proof: The storage proof ensures that the dataset is correctly stored in the BMHT and remains within the fixed number of leaves, preventing expansion. The BMHT root is cryptographically linked to the dataset using an HSMs-signed commitment:

$$\tau = \text{Sign}_{\text{HSMs}}(H_{\text{root}} \parallel N \parallel S), \quad H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau).$$

The CP provides:

$$\pi_{\text{storage}} = \{H_{\text{leaf}_1}, H_{\text{leaf}_2}, \ldots, H_{\text{leaf}_N}, H'_{\text{root}}\},$$

where each leaf is computed as $H_{\text{leaf}_i} = \text{Hash}(k_i \parallel E_i)$. The verifier (DO/CO) checks: $H'_{\text{root}} \stackrel{?}{=} \text{Hash}(H_{\text{root}} \parallel \tau)$ and reconstructs H_{root} from π_{storage} to verify its consistency. The proof is valid if the recomputed H_{root} matches the expected value, confirming that the storage structure has not been altered.

Membership Proof: The membership proof verifies the inclusion of a specific block D_i in the BMHT. The cloud provider provides the Merkle root $\pi_{\text{membership}}$ from the corresponding leaf H_{leaf_i} to the H_{root}, while the DO provides the expected root hash H_{root} for verification.: $\pi_{\text{membership}} = \{H_{\text{sibling}_1}, H_{\text{sibling}_2}, \ldots, H_{\text{sibling}_h}\}$, where $h = \lceil \log_2(N) \rceil$ is the height of the tree. The verifier performs the following steps: 1. Computes the root hash iteratively:

$$H^{(1)}_{\text{path}} = \text{Hash}(H_{\text{leaf}_i} \parallel H_{\text{sibling}_1}),$$

$$H^{(j+1)}_{\text{path}} = \begin{cases} \text{Hash}(H^{(j)}_{\text{path}} \parallel H_{\text{sibling}_{j+1}}), & \text{if left-child,} \\ \text{Hash}(H_{\text{sibling}_{j+1}} \parallel H^{(j)}_{\text{path}}), & \text{if right-child,} \end{cases} \tag{5.1}$$

for $j = 1, \ldots, h-1$. 2. Verifies: $H^{(h)}_{\text{path}} = H_{\text{root}}$. The proof is valid if only the computed root matches the DO's provided H_{root}.

Non-Membership Proof: The non-membership proof ensures that a queried key $k \notin \{k_1, k_2, \ldots, k_N\}$, leveraging the sorted property of BMHT leaves and the concepts of frontier and boundary sets [194]. Sentinel nodes $-\infty$ and $+\infty$ are introduced to handle edge cases where k is smaller than k_1 or larger than k_N. These are fixed, predefined virtual keys serving as placeholders at the BMHT boundaries [195].

Definitions and Proof Generation by CP:

- **Sentinel Nodes:** Predefined values are assigned to sentinel hashes:

  $$H_{-\infty} = \text{Hash}(\text{“-infinity”}), \quad H_{+\infty} = \text{Hash}(\text{“+infinity”}),$$

  ensuring they are publicly known and verifiable constants. Sentinel nodes $-\infty$ and $+\infty$ are included as the first and last leaves in the BMHT:

  $$\text{Leaves} = [H_{-\infty}, H_1, H_2, \ldots, H_N, H_{+\infty}].$$

- **Frontier Set (F):** The smallest subset of keys required to validate $k \notin \{k_1, \ldots, k_N\}$:

  $$F = \{k_{\text{pred}}, k_{\text{succ}}\}, \quad \text{where } k_{\text{pred}} < k < k_{\text{succ}}.$$

  For edge cases:

  $$k < k_1 \implies k_{\text{pred}} = -\infty, \quad k > k_N \implies k_{\text{succ}} = +\infty.$$

- **Boundary Set (B):** The Merkle hashes along the paths from $H_{\text{leaf}_{\text{pred}}}$ and $H_{\text{leaf}_{\text{succ}}}$ to the root: $B = \{\pi_{\text{pred}}, \pi_{\text{succ}}\}$, where π_{pred} and π_{succ} are the Merkle paths.
- **Proof Generation:** The CP identifies k_{pred} and k_{succ} such that $k_{\text{pred}} < k < k_{\text{succ}}$, then computes π_{pred} and π_{succ}.

Verification by the Verifier:

- **Check Key Order:** Verify: $k_{\text{pred}} < k < k_{\text{succ}}$. For edge cases, confirm: $k_{\text{pred}} = -\infty$ or $k_{\text{succ}} = +\infty$, ensuring sentinel nodes are correctly referenced.
- **Recompute Root from Boundary Set:** Use π_{pred} and π_{succ} to recompute the root:

$$H^{(j+1)}_{\text{path-pred}} = \begin{cases} \text{Hash}(H^{(j)}_{\text{path-pred}} \,\|\, H_{\text{sibling}_{j+1}}), & \text{if left-child,} \\ \text{Hash}(H_{\text{sibling}_{j+1}} \,\|\, H^{(j)}_{\text{path-pred}}), & \text{if right-child.} \end{cases}$$

  where $H^{(1)}_{\text{path-pred}} = H_{\text{leaf}_{\text{pred}}}$. Similarly, compute $H^{(h)}_{\text{path-succ}} = H_{\text{root}}$.
- **Validate Non-Membership:** The proof is valid if:

$$H^{(h)}_{\text{path-pred}} = H_{\text{root}}, \quad H^{(h)}_{\text{path-succ}} = H_{\text{root}},$$

  and: $k \notin \{k_{\text{pred}}, k_{\text{succ}}\}$. If $k = k_{\text{pred}}$, $k = k_{\text{succ}}$, or the paths fail to recompute H_{root}, the proof is invalid.

Verification is efficient, with storage proofs requiring $O(N)$ hash operations for root computation, while incremental updates rehash only affected parts, reducing complexity to $O(\log_2 N)$. BMHT leverages an HSM-signed commitment, enabling $O(1)$ verification by checking a single hash signature. Membership proofs operate in $O(\log_2 N)$, while non-membership requires verifying two paths, yielding $O(\log_2 N)$. Security relies on collision-resistant hash functions, ensuring data integrity, with adversarial forgery probability bounded by $\epsilon(k)$, where k is the security parameter.

5.5.3. DATA SHARING, SECURE ACCESS, ZERO-RESIDUALS OVERWRITING, AND GLOBAL CONSISTENCY

This phase ensures secure data sharing, controlled access, zero-residual overwriting, and verifiable deletion across CPs using HSMs, PKI, Secure Enclaves, and GMF.

Data Sharing and Encryption: The HSMs generate asymmetric key pairs for co-owners (COs) at the respective provider. For each CO_j, let: $\{PK_j, SK_j\} \leftarrow \text{KeyGen}_{\text{HSM}}(1^k)$, where PK_j and SK_j are the public

and private keys, respectively, and k is the security parameter. The main HSM in the organizer CP encrypts each block D_i for a CO using $E_i = \text{Enc}_{PK_j}(D_i)$ provided by their HSM. The encrypted blocks $\{E_1, E_2, \ldots, E_N\}$ are uploaded to CPs, ensuring only authorized COs with SK_j can decrypt: $D_i = \text{Dec}_{SK_j}(E_i)$. Their HSMs is responsible for decrypting the data.

Secure Access and Session Keys: Data access is restricted within a Secure Enclave initialized by the COs to prevent local copies. The HSMs receives encrypted data, initially secured using a PKI-based asymmetric key specific to the provider. The HSM decrypts the data using the corresponding private key: $D_i = \text{Dec}_{K_{\text{PKI}}}(C_{\text{stored}})$. To enable secure communication between the HSM and the enclave, the HSM generates a temporary session-specific symmetric key, known as the session key: $K_{\text{session}} \leftarrow \text{KeyGen}_{\text{HSM}}(1^k)$. The decrypted data is then re-encrypted with this session key: $C_i = \text{Enc}_{K_{\text{session}}}(D_i)$. This allows the enclave to securely access the data without exposing the provider's PKI private key. The session key ensures confidentiality during transmission and access within the cloud. Once the access session is complete, the HSM securely destroys K_{session}, ensuring it cannot be recovered or reused: $\text{Destroy}(K_{\text{session}})$. This mechanism ensures that long-term PKI keys remain protected while enforcing session-based encryption for temporary access to on-cloud.

Zero-Residuals Overwriting: To ensure secure and provable data deletion, old data D_{old} is overwritten with D_{new} using a structured three-phase process based on Pseudorandom Functions (PRFs) and Pseudorandom Permutations (PRPs) [196], following NIST SP 800-88 guidelines for secure sanitization [190]. Since residual patterns may persist due to storage behavior or compression artifacts, statistical randomness tests, as recommended by NIST SP 800-22 [197], validate that D_{new} is indistinguishable from a truly random sequence, preventing adversarial recovery and maximizing entropy. Let:

- $f_k(x) : \{0,1\}^k \to \{0,1\}^l$ be a PRF, where k is the security parameter and l is the output length.
- $\pi_k(x), \mu_k(x) : \{0,1\}^k \to \{1, 2, \ldots, n\}$ be PRPs, defining secure data transformations.
- $H(\cdot)$ be a cryptographic hash function ensuring integrity.

The secure overwriting process consists of the following three steps:

1. PRF Masking: The old data is masked using a PRF to obscure its original structure:

$$D_1 = D_{\text{old}} \oplus f_k(H(D_{\text{old}})).$$

2. Bit Reversal: The masked data undergoes bit-wise reversal followed by an additional PRF-based re-masking:

$$D_2 = \text{ReverseBits}(D_1) \oplus f_k(H(D_1)).$$

3. Entropic Shuffling: A PRP is applied to shuffle the bits, increasing entropy and preventing pattern recognition:

$$D_{\text{new}} = \pi_k(D_2).$$

The PRP also assigns D_{new} a new storage position:

$$P_{\text{new}} = \pi_k(P_{\text{old}}).$$

The Bounded Merkle Hash Tree (BMHT) root is updated accordingly per each cycle: $H_{\text{leaf}_{\text{new}}} = \text{Hash}(k_{\text{new}} \parallel D_{\text{new}})$

Zero-Residuals Analysis: Let D_{new} be the overwritten data block $(\{d_1, d_2, \ldots, d_n\})$. If D_{new} is large, partition it into m segments:

$$D_{\text{new}} = \bigcup_{i=1}^{m} \Delta_i.$$

Each segment Δ_i is set to 512 KB, ensuring a balance between computational efficiency and randomness detection, as NIST SP $800-22$ recommends [197]. To validate that D_{new} is statistically indistinguishable from a uniformly random sequence, we assess its randomness using J statistical tests. Define a function $T : \{0,1\}^n \to \mathbb{R}$, which computes a numerical statistic $T(X)$ from a binary sequence X by evaluating frequency balance, run lengths, and pattern occurrences. For the j-th test applied to segment Δ_i, let: $T_{i,j}^{\text{obs}} = T(\Delta_i)$, and let $T_{i,j}^{\text{rand}}$ denote the expected value under true randomness. The corresponding p-value is then calculated as:

$$p_{i,j} = \Pr\left[T_{i,j}^{\text{rand}} \geq T_{i,j}^{\text{obs}}\right].$$

To ensure randomness validation, we compute the average p-value $\Lambda(\Delta_i)$:

$$\Lambda(\Delta_i) = \frac{1}{J}\sum_{j=1}^{J} p_{i,j}, \quad \Lambda_{\text{pass}} = \mathbb{I}(\forall j, p_{i,j} \geq \alpha).$$

The empirical threshold $\alpha = 0.05$, widely accepted in statistical cryptographic applications, determines segment acceptance:

$$D(\Delta_i) = \begin{cases} \text{Pass}, & \text{if } p_{i,j} \geq \alpha \quad, \forall j = 1, \ldots, J, \\ \text{Fail}, & \text{otherwise.} \end{cases}$$

The statistical validation integrates three key randomness tests:

(1) Frequency Test, ensuring a balanced distribution of bits using:

$$S_n = \sum_{k=1}^{n} (2x_k - 1).$$

(2) Run Test, verifying that transitions between '0's and '1's and their standard deviation (σ) conform to expected distributions via:

$$Z_{\text{run}} = \frac{|V_n - E[V_n]|}{\sqrt{\sigma^2}}, \quad V_n = \text{number of bit sequences.}$$

(3) Pattern Test, detecting repetitive structures through Approximate Entropy (ApEn) of entropy measure (Φ), computed as:

$$\text{ApEn} = \Phi_m - \Phi_{m+1}, \quad \Phi_m = \frac{1}{n-m+1} \sum_{k=1}^{n-m+1} \log P_k(m),$$

where $P_k(m)$ refers to empirical frequency. If any segment Δ_i fails (i.e., $D(\Delta_i) = \text{Fail}$), the re-randomization process (adjusting PRF/PRP parameters) is applied iteratively until all segments satisfy $p_{i,j} \geq \alpha$. This guarantees that no residual structure remains, ensuring complete and provable data deletion. The choice of $\alpha = 0.05$ balances Type I and Type II errors, minimizing false positives and negatives, as recommended by NIST.

Irreversibility Guarantee: Reconstructing D_{old} from D_{new} would require inverting the PRF, PRP, and bit reversal, which is computationally infeasible under standard cryptographic assumptions:

$$D_{\text{old}} = f_k^{-1}\Big(\text{ReverseBits}^{-1}\big(\pi_k^{-1}(D_{\text{new}}) \oplus f_k(H(D_1))\big)\Big).$$

Overwriting Proof with Storage Commitment: The proof ensures that D_{old} at position P_{old} is securely replaced by D_{new} at P_{new}, where $P_{\text{old}}, P_{\text{new}} \in \mathbb{N}$, while maintaining strict storage enforcement: $\pi_{\text{overwrite}} = \pi_{\text{membership}} \cup \pi_{\text{non-membership}}$. After overwriting, the updated root must satisfy the fixed storage commitment, ensuring integrity: $H'_{\text{root}} = \text{Hash}(H_{\text{root}} \parallel \tau)$. To verify the proof, the verifier checks inclusion by ensuring $H^{(h)}_{\text{path-new}} = H_{\text{root}}$, confirming that D_{new} is correctly inserted. Exclusion is verified by ensuring that $k_{\text{pred}} < k_{\text{old}} < k_{\text{succ}}$ holds, along with the recomputed paths satisfying $H^{(h)}_{\text{pred}} = H^{(h)}_{\text{succ}} = H_{\text{root}}$, guaranteeing that D_{old} is removed. The storage enforcement is validated using $\text{Verify}_{\text{HSM}}(\tau, H_{\text{root}}, N, S)$. The proof is accepted if all checks hold:

$$\pi_{\text{overwrite}} = \begin{cases} 1, & \text{if all checks are satisfied,} \\ 0, & \text{otherwise.} \end{cases}$$

This ensures that permuted overwriting respects the limited storage constraints (N, S) and prevents residual data from persisting.

Global Proof Synchronization: The GMF aggregates BMHT roots H_{root} from all CPs, ensuring global consistency. The aggregated root is periodically (i.e., Based on configuration) recomputed and stored in the

organizer CP's HSMs: $H_{\text{GMF}} = \text{Hash}(H_{\text{root}_1} \parallel H_{\text{root}_2} \parallel \ldots \parallel H_{\text{root}_m})$, where M is the total number of CPs. This guarantees consistent verification across asynchronous environments.
Encryption operations scale linearly with the number of blocks, as asymmetric encryption for data sharing requires $O(N)$ computations. Overwriting via PRF/PRP occurs per block, maintaining $O(N)$ complexity. Verifying overwriting proofs, including membership and non-membership, follows $O(\log_2 N)$ complexity. The GMF aggregates BMHT roots from M providers into a single-level Merkle structure. Construction requires hashing all M roots, yielding $O(M)$, while verification is constant-time at $O(1)$, ensuring minimal overhead. The security of overwriting and GMF proofs relies on collision-resistant hashes, PRFs, PRPs, and bit reversals, ensuring irreversibility with adversarial success probability $\epsilon(k)$.

5.6. FORMAL ANALYSIS

Let $D = \{d_1, d_2, \ldots, d_n\}$ be the dataset stored in the BMHT, where each $d_i = (k_i, D_i)$ is a key-value pair with k_i as the unique key and D_i as the encrypted data block. The BMHT root H_{root} cryptographically commits to D, ensuring integrity and verifiability.

Theorem 5.6.1 (Correctness)**.** *A valid proof π generated by the CP enables the verifier (DO or CO) to verify storage, membership, non-membership, and overwriting operations (Def. 5.2.3). Formally, correctness holds if, for negligible $\epsilon(k)$:*

$$\Pr[V_\pi(D, H_{root}, \pi) = 1] \geq 1 - \epsilon(k)$$

Where V_π is the verification function (i.e., mentioned §5.2.3), and the probability is taken over the randomness of the cryptographic hash function and pseudorandom permutations [196].

Proof. Correctness follows from the collision resistance of the cryptographic hash function [198]. Any adversary $\mathcal{A}$ generating an incorrect proof π' must construct a falsified root H'_{root} such that: $V_\pi(\pi') = 1$ but $H'_{\text{root}} \neq H_{\text{root}}$. This requires a hash collision, which occurs with negligible probability: $\Pr[\text{Hash Collision}] \leq \epsilon(k)$.
For membership, the verifier checks if $\pi_{\text{membership}}$ correctly reconstructs H_{root}. Forging a valid proof requires breaking preimage resistance:

$$\Pr[\text{Forge Membership Proof}] \leq \epsilon(k).$$

For non-membership proof ensures that a queried key k is absent by proving it lies between two adjacent committed keys k_{pred} and k_{succ}:

$k_{\text{pred}} < k < k_{\text{succ}}$. The prover supplies Merkle proofs π_{pred} and π_{succ}, allowing the verifier to recompute H_{root} and confirm:

$$H^{(h)}_{\text{path-pred}} = H_{\text{root}}, \quad H^{(h)}_{\text{path-succ}} = H_{\text{root}}.$$

Since BMHT enforces key order, falsifying non-membership requires forging H_{root} or producing a collision, both of which occur with negligible probability:

$$\Pr[\text{Forge Non-Membership Proof}] \leq \epsilon(k).$$

For overwriting, correctness ensures that a valid proof confirms the old value is replaced with the new one. The BMHT updates maintain consistency:

$$V_{\pi_{\text{overwrite}}}((k_{\text{old}}, D_{\text{old}}), (k_{\text{new}}, D_{\text{new}}), H_{\text{root}}, \pi_{\text{overwrite}}) = 1.$$

Thus, correctness holds under collision-resistant hashes, BMHT binding, and PRF/PRP security, ensuring integrity and preventing forgery. □

Theorem 5.6.2 (Security)**.** *For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, a malicious cloud provider attempting to forge proofs, introduce inconsistencies across the GMF, collude with other CPs, or perform partial overwriting will be detected with high probability (Def. 5.2.3). Let π be a valid proof for storage, membership, non-membership, or overwriting, and let π' be a falsified proof. Let $\mathcal{H}^{(i)}_{root}$ be the BMHT root of provider i, and $\mathcal{H}_{GMF}$ be the global Merkle root aggregating all providers. The probability that $\mathcal{A}$ successfully forges π' or introduces inconsistencies in GMF is bounded by:*

$$\Pr\left(V_{\pi}(\pi') = 1 \;\middle|\; (\pi' \neq \pi) \vee \exists\, i \in \{1, \ldots, M\} : \mathcal{H}^{(i)}_{BMHT} \notin \mathcal{H}_{GMF}\right) \leq \epsilon(k)$$

where $\epsilon(k)$ is negligible in the security parameter k.

Proof. A successful attack requires constructing a false proof π' such that: $H'_{\text{root}} = H_{\text{root}}$, for $D' \neq D$. Since BMHT relies on collision-resistant hash functions with preimage resistance, the probability of a successful forgery is negligible:

$$\Pr[\text{Forge}(\pi')] \leq \epsilon(k).$$

The verifier ensures proof integrity using the BMHT root: $H_{\text{root}} = \text{Hash}(H_{\text{leaf}_1} \| \cdots \| H_{\text{leaf}_N})$. Ensuring that any unauthorized modification is detected. The root is cryptographically bound to data and signed via HSMs: $\tau = \text{Sign}_{\text{HSM}}(H_{\text{root}} \| N \| S)$, breaking τ would require forging a digital signature, which happens with negligible probability. The GMF enforces cross-provider integrity through: $H_{\text{GMF}} = \text{Hash}(H_{\text{root}_1} \| H_{\text{root}_2} \| \cdots \| H_{\text{root}_M})$. Any inconsistency across

providers is cryptographically detectable but not preventable, as a malicious CP may refuse to provide valid proof.

Confidentiality via HSMs and Secure Enclaves. HSMs enforce encryption, protecting data via PKI with *IND-CCA2* security; all encryption and decryption occur inside the HSM, so CPs never access keys or plaintext. Inside Secure Enclaves, data is processed under HSM-provisioned session keys using *IND-CPA*-secure symmetric encryption, ensuring CPs cannot infer plaintext. Even if CPs collude, they cannot extract data, as each HSM independently enforces key isolation.

Resistance to Colluding CPs. CPs cannot decrypt or share meaningful data due to provider-specific HSM keys, end-to-end encryption, and ephemeral enclave keys. Since BMHT enforces fixed storage allocation, any modification outside the defined storage structure is detected.

Partial Overwriting Detection. The BMHT ensures that overwriting follows a verifiable sequence:

$$V_{\pi_{\text{overwrite}}}((k_{\text{old}}, D_{\text{old}}), (k_{\text{new}}, D_{\text{new}}), H_{\text{root}}, \pi_{\text{overwrite}}) = 1.$$

Proof verification detects any deviation from complete overwriting, enforcing storage integrity. □

While BMHT and GMF detect forgery and inconsistencies, system-level mechanisms handle secure deletion.

Zero-residual Overwritten data undergoes a structured three-phase process at the system-level:

$$F(D) = \pi_k\Big(\text{ReverseBits}\Big(D \oplus f_k(H(D))\Big)\Big).$$

Recovering D_{old} from D_{new} is infeasible under PRF and PRP security [199, 200]: $\Pr\big[\mathcal{A} \text{ inverts } F\big] \leq \epsilon(k)$. Overwriting is a storage-level operation, not a cryptographic primitive, and provides no standalone guarantee. It complements cryptographic erasure, with empirical randomness tests validating it as a standards-aligned measure.

Mitigating Malicious COs via Code Enforcement. While our work focuses on trusted-but-forgetful COs, it is possible to impede malicious COs by restricting enclave execution to DO-approved code. Each enclave instance generates a code measurement $M_{\text{code}} = H(\texttt{code})$, where H is a cryptographic hash over the enclave binary. The DO approves a specific measurement M_{DO} corresponding to the authorized code. During remote attestation, the enclave sends $(M_{\text{code}}, \mu_{\text{att}})$, where μ_{att} is a signed attestation message from enclave hardware. The HSM releases the decryption key K only if the attestation is valid and matches the DO-approved measurement:

$$\text{KeyRelease}_{\text{HSM}} \iff \text{VerifyAttest}(\mu_{\text{att}}) \wedge (M_{\text{code}} = M_{\text{DO}}).$$

If a CO executes modified code, verification fails and the key is withheld: $\Pr[\text{KeyAccess} \mid M_{\text{code}} \neq M_{\text{DO}}] = 0$. Let $\mathcal{A}_{\text{CO}}$ be an adversarial co-owner. The probability of recovering sensitive data D_i without access to K is bounded by:

$$\Pr[\mathcal{A}_{\text{CO}} \to D_i] \leq \Pr[\mathcal{A}_{\text{CO}} \to K] + \Pr[\text{Break Enclave}] \leq \epsilon(k).$$

Malicious COs cannot recover D_i without running DO-approved code, enforced via attestation-based mitigation.

5.7. RESULTS AND DISCUSSION

To evaluate the practicality of our proposed framework, we developed a prototype implementation in Python (3.9) that supports verifiable storage, secure deletion, and proof generation for key operations—including BMHT, GMF, and zero-residual overwriting. To deploy the system, we used the AWS cloud with CloudHSM [201], Nitro Enclaves [202], and local proof verification to ensure provable data deletion with zero residuals and verifiability. CloudHSM (`hsm1.medium`) instances in `eu-north-1a` and `eu-north-1b` zones provide FIPS 140 − 2 Level 3 compliance [193] for secure key lifecycle management, handling key generation, encryption, decryption, and controlled destruction. AWS KMS integrates with CloudHSM via private VPC endpoints for key management [203]. BMHT roots are dynamically generated using AWS Lambda [204], triggered by S3 (Standard Class) storage modifications [205], with cryptographic commitments stored in DynamoDB [206] for proof verification. The computing setup includes an EC2 `t4g.micro` instance for HSM operations [207], a `c6i.xlarge` Nitro Enclave for secure decryption and proof generation [202], and a local MacBook with a 2.3 GHz Intel Core *i*5 and 8 GB RAM for independent proof verification. CloudWatch [208] monitors performance metrics such as CPU utilization, cost, throughput, and request latency. This section details (*i*) Computational Complexity and (*ii*) Experimental Results.

5.7.1. COMPUTATIONAL COMPLEXITY

PROOF GENERATION TIME.

Proof generation occurs at the *Cloud Provider (CP)* level upon any storage modification event (e.g., data overwriting, updates). When a change is detected, the BMHT root hash is recomputed and stored as a cryptographic commitment in *Amazon DynamoDB* as a key-value record. Since BMHT maintains a structured and bounded design, proof generation time remains consistent per overwrite event. The *event-driven nature* of proof updates ensures that each modification

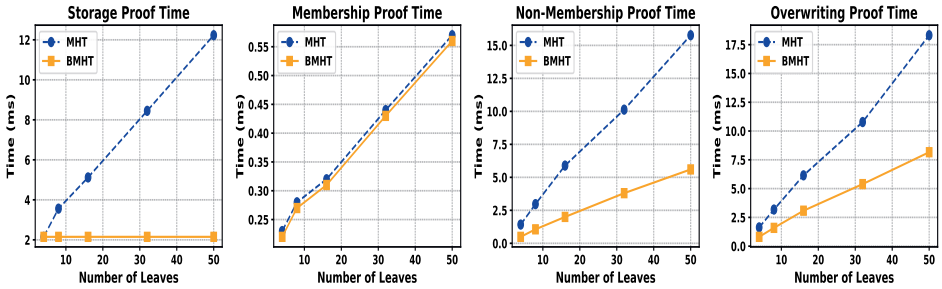


Figure 5.3: Comparison of BMHT and Traditional MHT Verification Times for Different Proofs.

triggers a single proof computation without additional overhead, making generation time fixed per event.

5

PROOF VERIFICATION TIME (FIGURE 5.3).

This experiment evaluates the verification time for storage, membership, non-membership, and overwriting proofs in both MHT and BMHT locally from the user's end. *While proofs are verified locally by default, our system also supports enclave-side and hybrid verification, where enclaves attest to proof validity (e.g., success/failure) to reduce local storage and bandwidth overhead.* Table 5.2 highlights BMHT's efficiency, particularly in non-membership and overwriting computational proof complexity. For storage proof, BMHT achieves constant-time storage frequent verification ($O(1)$), maintaining 2.15 ms across all leaf sizes due to cryptographic enforcement by the HSMs, which eliminates the need for full-tree recomputation. While MHT requires full-tree recomputation at all times, increasing from 2.15 ms at $N = 4$ to 12.23 ms at $N = 50$. Membership proofs are nearly identical, as both follow a Merkle path to the root ($O(\log N)$), yielding 0.56 ms for BMHT and 0.57 ms for MHT at $N = 50$. For non-membership proofs, BMHT efficiently finds predecessor-successor nodes in $O(\log N)$, reducing verification time to 5.60 ms at $N = 50$ compared to MHT's 15.77 ms, which requires scanning all leaves. Similarly, BMHT's overwriting proofs leverage its structured design, achieving 8.16 ms at $N = 50$ versus MHT's 18.32 ms.

5.7.2. EXPERIMENTAL RESULTS

Our experiments are based on: (*i*) Overwriting Time, (*ii*) HSM Performance, (*iii*) Secure Eclave Analysis, (*iv*) Scalability Assessment, (*v*) Cost Estimation, and (*vi*) Latency Analysis.

Table 5.2: Comparison of BMHT and MHT Verification Time Complexities.

Proof Type	BMHT Complexity	MHT Complexity
Storage	$O(1)$	$O(N)$
Membership	$O(\log N)$	$O(\log N)$
Non-Membership	$O(\log N)$	$O(N)$
Overwriting	$O(\log N)$	$O(N)$

ZERO-RESIDUALS OVERWRITING TIME.

We evaluate the computational cost of block permutation across varying sizes on the cloud, with results shown in Figure 5.4. The experiment measures overwriting time for a fixed total data size while varying only the block size, ensuring execution time differences reflect the impact of block size alone. As block size increases, permutation time rises due to the three-step zero-residuals overwriting process: PRF masking, bit reversal, and entropic shuffling—each operating on all data symbols to maximize entropy. Larger blocks require more randomized accesses during PRP-based shuffling (via SHA-256 bit [209]), increasing computational overhead as the number of elements to be repositioned scales with block size. While PRF masking and bit reversal introduce minimal latency, PRP shuffling dominates execution time due to its intensive randomization, ensuring data patterns are effectively obfuscated. This aligns with NIST SP 800-88 media sanitization guidelines [190], which mandate cryptographic erasure to achieve zero-residual guarantees. These results highlight the role of block size in secure deletion, as larger blocks require more processing time to achieve complete data obfuscation, ensuring overwritten data remains computationally irrecoverable. The elapsed time accounts solely for the overwriting process, excluding the statistical test used for the zero-residual guarantee.

HSM PERFORMANCE (FIGURE 5.5).

We evaluate the performance of AWS CloudHSM for cryptographic operations, utilizing AES-256-CBC for symmetric encryption and RSA-2048-OAEP for asymmetric encryption [210, 211]. The experiment measures execution time, CPU utilization, memory consumption, and throughput. The CloudHSM cluster consists of two nodes, with operations executed in a single-threaded workload to isolate computational overhead and accurately assess the performance of hardware-backed cryptographic processing. The results demonstrate the effectiveness of CloudHSM in securely handling both symmetric and asymmetric cryptographic operations while offloading the computational burden from general-purpose CPUs.

a) *Time Measurement:* Encryption and decryption were measured

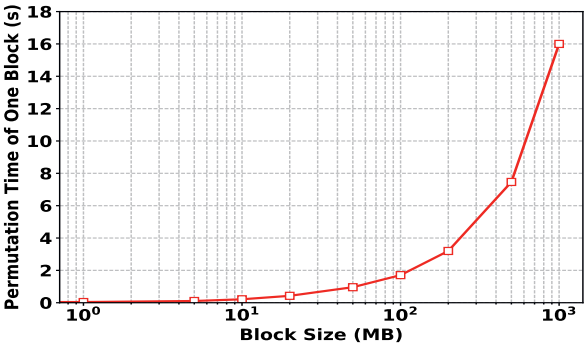


Figure 5.4: Permuted Overwriting Time for One Block on Different Block Size.

across varying data sizes, showcasing the efficiency of CloudHSM in securely handling cryptographic workloads. AES-256 encryption maintained consistently low latency, increasing from 0.01s for small blocks to 1.9s for large datasets, benefiting from hardware acceleration within the HSM. Decryption followed a similar pattern, peaking at 2.85s. RSA-2048 encryption, expectedly more computationally intensive due to its reliance on modular exponentiation, exhibited a steeper increase in processing time, from 0.2s for small blocks to 9.2s for larger ones. RSA decryption, which requires complex mathematical computations, peaked at 16.0s. Despite the inherent differences between AES and RSA, CloudHSM effectively manages these operations, providing dedicated hardware acceleration for cryptographic techniques and ensuring security and compliance without exposing private keys outside the HSM.

b) *CPU Usage:* CPU utilization during encryption and decryption reflects CloudHSM's ability to offload cryptographic workloads. AES-256 encryption showed efficient CPU usage, ranging from 3% to 27%, benefiting from optimized hardware. RSA-2048 exhibited higher CPU utilization, reaching up to 53%, as expected due to its computational intensity. However, CloudHSM ensures that even RSA operations, which are typically CPU-intensive, do not overload the primary system, demonstrating the advantage of hardware-backed cryptographic processing. This CloudHSM-secured environment prevents unauthorized key exposure while efficiently executing encryption and decryption processes.

c) *Memory Usage:* Memory consumption was analyzed to assess the impact of CloudHSM-backed encryption/decryption. AES-256 maintained a moderate memory footprint, increasing from approximately 976 KB to around 10.5 MB as block sizes grew. RSA-2048, requiring

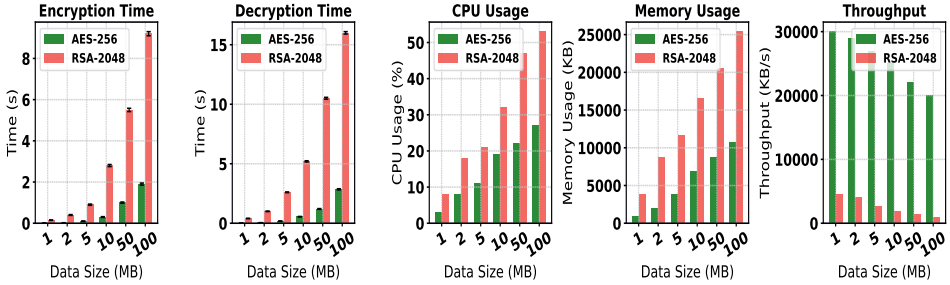


Figure 5.5: HSM Performance Evaluation for AES-256-CBC and RSA-2048-OAEP: Comparison of Encryption/Decryption Time, CPU Usage, Memory Consumption, and Throughput.

larger buffers for cryptographic calculations, exhibited higher memory usage, peaking at roughly 25 MB for the largest data sizes. Despite this, CloudHSM ensures memory efficiency by isolating cryptographic workloads from the central system, mitigating security risks associated with software-based key handling.

d) *Throughput:* It is measured in KB per second and highlights the efficiency of CloudHSM for several operations performed per unit of time. AES-256 achieved high throughput, starting at 30,000 KB/s for small blocks and decreasing to 20,000 KB/s as data size increased, demonstrating its suitability for large-scale data encryption. RSA-2048, while inherently slower due to its asymmetric nature, achieved a throughput of 4,500 KB/s initially, declining to 1,000 KB/s for the most significant blocks. These results confirm that CloudHSM handles symmetric and asymmetric encryption effectively, providing dedicated acceleration for high-security applications requiring strong cryptographic guarantees.

SECURE ENCLAVE ANALYSIS (FIGURE 5.6).

We analyze the performance of Nitro Enclaves in provable co-owned data deletion within multi-cloud environments. Unlike Standard EC2 instances, which optimize for general-purpose computing, Nitro Enclaves provide hardware-backed isolation, ensuring sensitive data remains confined to the cloud during decryption and processing. This evaluation focuses on decryption time, startup overhead, CPU utilization, and memory usage, balancing security guarantees with computational efficiency.

a) *Decryption Time:* The direct integration of Nitro Enclaves with an HSM ensures cryptographic operations occur securely without exposing key material to the host EC2 instance. However, this secure execution introduces a 30% increase in decryption (i.e., data retrieval) latency

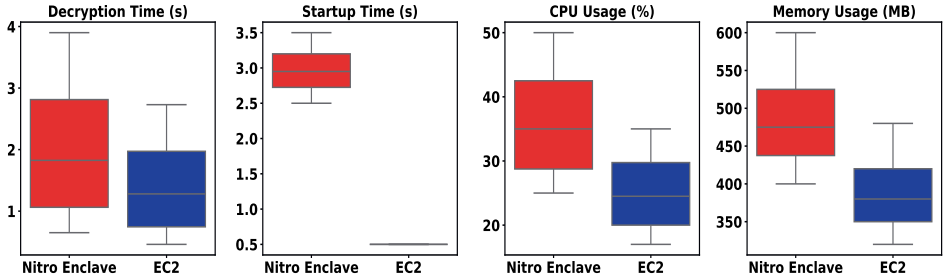


Figure 5.6: Nitro Enclave vs. Standard EC2 Performance in Decryption, Startup, CPU, and Memory Usage.

compared to Standard EC2, primarily due to attestation overhead and HSM-access constraints. Experimental results show decryption times of 0.65s for 5MB and 3.90s for 35MB, compared to 0.46s and 2.73s on Standard EC2. While EC2 benefits from direct CPU/memory access, Nitro ensures decryption keys remain isolated, making it essential for EHDS-compliant deletion.

b) *Startup Overhead:* Nitro Enclaves require between 2.5–3.5 seconds for initialization, significantly higher than EC2's near-instant execution. This is due to memory carving and remote attestation, which validate the enclave's integrity before cryptographic operations begin. Direct HSM integration does not introduce additional I/O bottlenecks; key derivation and access authentication add minor delays. The startup cost is justified in ensuring verifiable, tamper-proof deletion without exposing decryption operations to threats.

c) *CPU Usage:* Nitro Enclaves exhibit 25%–50% higher CPU utilization than EC2 due to their dedicated execution model. Secure memory handling and enclave-restricted cryptographic operations increase computational overhead. Results indicate 30% CPU usage for 10MB blocks and 50% for 35MB, whereas EC2 optimizes workload distribution, maintaining usage below 35%. Despite this, eliminating software-based encryption layers in Nitro ensures that cryptographic operations remain isolated from host-side attacks.

d) *Memory Usage:* Nitro Enclaves operate with fixed, pre-allocated memory blocks, preventing unauthorized access but leading to higher memory usage. Compared to EC2's 320MB–480MB dynamic allocation, Nitro reserves 400MB–600MB per session, ensuring no decrypted data persists beyond execution. Direct HSM integration optimizes memory efficiency by reducing unnecessary key storage, but the overall memory footprint remains larger due to isolation.

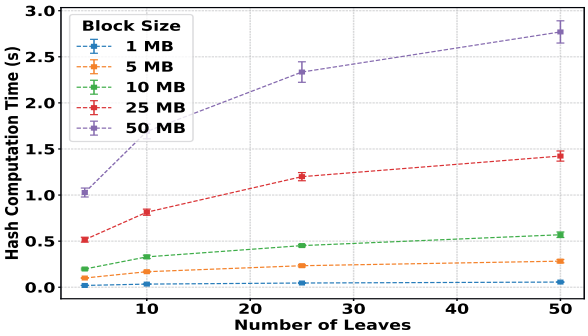


Figure 5.7: Scalability Analysis of BMHT: Hash Computation Time vs. Block Size and Number of Leaves.

SCALABILITY ASSESSMENT.

Scalability in the proposed scheme is evaluated based on the Bounded Merkle Hash Tree (BMHT) structure and the Global Merkle Forest (GMF) verification model. The BMHT experiment considers varying tree sizes $(4, 10, 25, 50$ leaves) and different block sizes $(1, 5, 10, 25, 50$ MB) to examine their impact on proof verification and hash computation time needed. *The proposed scheme scales linearly with data size for core operations and logarithmically for verification.*

BMHT: Block Size and Hash Computation (Figure 5.7). The BMHT structure follows a hierarchical Merkle tree with logarithmic proof verification complexity, $O(\log_2 N)$. Since BMHT stores only fixed-size hashes, verification time depends on the number of leaves, not block size. However, larger blocks increase hashing overhead. Experimental results show the minimal impact of block size on verification, while hashing a 1MB block remains under 10ms, and a 50MB block exceeds 500ms at 50 leaves. As BMHT operates solely on hashes, larger block sizes do not affect verification efficiency.

GMF: Constant-Time Verification. The Global Forest (GMF) consolidates multiple BMHT root hashes into a single-level Merkle structure. Since the GMF only stores root hashes, verification remains constant-time, i.e., $O(1)$, regardless of the number of BMHT leaves. This ensures that checking the integrity of a stored BMHT root remains computationally lightweight and efficient, making the GMF suitable for global proof consistency without additional computational overhead.

COST EXECUTION (TABLE 5.3).

To estimate the operational cost of enforcing provable co-owned data deletion, we approximate a total expenditure of $430–$460

Table 5.3: AWS Service Cost Estimation for Secure Co-Owned Data Deletion Adoption.

Service	Operations Performed	Usage Unit	Actor
CloudHSM	Cryptographic operations	$1.45/hr/HSM	Trusted Hardware
	Signing and key operations	Included in hourly cost	Trusted Hardware
Nitro Enclave (c6i.xlarge)	Secure execution for decryption	$0.18/hr	Trusted Hardware
AWS Lambda	Compute execution (pay-per-use)	$0.20/1M requests	Provider
	Execution time (GB-s)	$0.00001667/GB-s	Provider
Amazon DynamoDB	Read capacity units (RCUs)	$0.000125/RCU	Provider
	Write capacity units (WCUs)	$0.00065/WCU	Provider
	Storage cost	$0.25/GB-month	Provider
EC2 Standard (t4g.micro)	Baseline compute for orchestration	$0.0084/hr	Provider
S3 Standard	Data storage	$0.023/GB-month	Owners/Provider
	Data transfer (out to Internet)	$0.09/GB	Owners/Co-Owners
VPC & Networking	Secure enclave-HSM communication	Free (intra-region)	Provider
AWS PrivateLink	Secure API access within VPC	$0.01/hr + $0.01/GB	Provider
Other Charges	AWS service fees and tax	Approx. 10% of total	Provider

over seven days, distributed across multiple AWS services. The CloudHSM cluster, consisting of two HSM instances, accounted for $255.80, managing the key generation, encryption, decryption, and controlled key destruction. The dual-HSM setup ensured redundancy, high availability, and compliance with cryptographic security policies. Execution on a `c6i.xlarge` enclave instance incurred $114.24 for 210 sessions, each averaging 1.5 hours. Costs stemmed from decryption, proof generation, and attestation overhead.

Nitro Enclaves lack persistent key storage, necessitating CloudHSM for hardware-backed security, NIST FIPS 140-2 compliance, and controlled key destruction. Matching dual-HSM capabilities requires upgrading to at least `c6i.metal` at $5.94/hr, making CloudHSM the more cost-effective option. EC2 instances (`t4g.micro`) used for orchestration totaled $4.74 over seven days. VPC networking costs amounted to $1.11 for encrypted transfers between Nitro Enclaves and CloudHSM, ensuring secure, direct communication. AWS KMS contributed $0.66 for API-based key provisioning and signing, enforcing controlled key access within the enclave. Due to the pay-per-use model, S3, Lambda, and DynamoDB costs were minimal. A single stored file kept S3 costs low, Lambda executed only on data updates, and DynamoDB incurred costs only for the proof storage

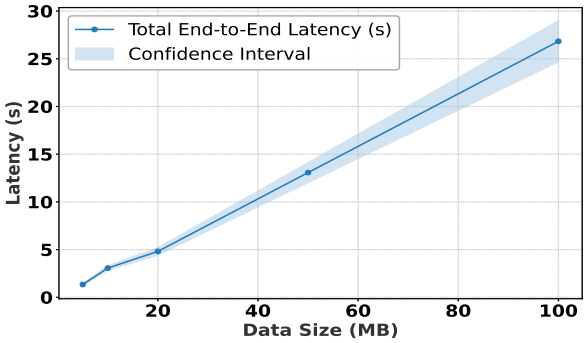


Figure 5.8: End-to-End Latency Analysis.

generation upon modifications. For comparison, a plaintext setup costs <$10/week, while an enclave-only alternative with `c6i.metal` around $1000/week-highlighting the efficiency of our approach at nearly half the cost. Such costs are practical for regulated compliance-heavy sectors like healthcare and finance, where organizations routinely spend millions annually on compliance (e.g., >$7.5M/year for U.S. hospitals [212]), making our verifiable, standards-compliant solution cost-effective with a lower total cost of ownership.

LATENCY ANALYSIS (FIGURE 5.8).

This experiment measures the end-to-end latency of secure co-owned data deletion, considering HSM encryption, BMHT proof generation, secure enclave decryption, and proof sharing. Confidence intervals (±8%) account for HSM acceleration and enclave constraints. HSM encryption ensures minimal overhead with consistently low latency due to hardware acceleration. BMHT proof generation remains efficient with a 4-leaf structure, keeping verification time under one second. Secure enclave decryption, performed in-memory within AWS Nitro Enclaves, introduces slight non-linearity due to processing constraints, particularly for large datasets. Data sharing adds minimal communication latency. Results show that HSM encryption maintains low latency, BMHT proof generation is efficient, and enclave decryption incurs higher latency but remains within practical limits.

5.8. RELATED WORK

Data link removal, encryption, and overwriting are the three main strategies that now constitute secure data deletion techniques in cloud service contexts. Although these techniques cover several data deletion

issues, their shortcomings are especially noticeable regarding co-owned data erasure in multi-cloud scenarios. The link removal strategy involves severing the logical connection between a data file and its associated file system [167, 169]. Kavous et al. [168] proposed an approach that achieves logical deletion by removing links, which prevents unnecessary access to obsolete data. This technique additionally incorporates a delayed deletion mechanism to reduce the possibility of unintentional deletions. The actual data information, however, is still on the storage device and can be recovered using forensic methods like disk scanning. This residual data renders the link removal strategy unsuitable for securely erasing co-owned data, particularly in multi-cloud scenarios where physical deletion is crucial for ensuring trust.

The data encryption strategy renders data inaccessible by encrypting and deleting the associated encryption key. Cachin et al. [171] introduced a policy-based scheme that removes attributes and protection classes to enforce key deletion. Darwish et al. [56, 87] extended this concept with key-decay schemes: one corrupts the ephemeral keys after use without requiring ephemeral storage, while another integrates audience-based expiration in order to render the data inaccessible after its expiry period. Jaeheung et al. [170] proposed a method where encryption keys are stored with metadata, and deletion involves overwriting the key's disk location, making the ciphertext inaccessible. Feng et al. [53] presented a data deletion verification scheme utilizing a Trusted Platform Module (TPM). The TPM securely manages encryption keys, generates signatures to verify deletion, and removes keys after expiration, ensuring data remains inaccessible. While this method supports public authentication, the TPM's limited storage capacity restricts scalability. Although effective for logical deletion, encryption-based approaches depend heavily on the secure removal of keys, leaving ciphertext on storage media. Residual ciphertext poses risks for co-owned data in multi-cloud environments, where deletion must ensure both logical and physical removal. Key management across providers adds complexity, requiring secure verification to prevent unauthorized access.

The overwriting strategy eliminates data by replacing it with random or predefined patterns to ensure irretrievability. Yuchuan et al. [172] introduced a rule transposition algorithm (RTA) to optimize the costs of generating and transferring random data for overwriting, improving efficiency. Daniele and Gene [174] proposed PoSE-s, which securely overwrite storage media with random patterns to prevent data recovery. Tian et al. [175] proposed SEAD-OO, a secure, assured deletion scheme combining orderly overwriting with ciphertext-policy attribute-based encryption to enable fine-grained access control and data sharing. The scheme ensures both physical and logical deletion of ciphertext, leveraging blockchain for traceability and deletion validation. Singh et al. [173] highlighted challenges with Solid State Drives (SSDs), where

wear-leveling techniques can prevent complete overwriting, leaving traces of deleted data vulnerable to recovery. Existing overwriting approaches focus on single-owner data and lack mechanisms for zero-residual overwriting and verifiable co-owned data deletion, making them inadequate for shared data in multi-cloud environments due to cross-provider gaps requiring verifiable proofs for data removal.

Recent studies have extended the scope from deletion alone to broader lifecycle management. Scope et al. [213] proposed a compliance-oriented framework for retention and purging in databases and backups. However, their system lacks support for physical deletion, verifiability, or multi-cloud coordination. Gao et al. [214] introduced SecCask, a system that leverages TEEs for secure data analytics pipeline evolution. While it ensures confidentiality during processing, it does not provide mechanisms for secure sharing, consistent updates, or deletion guarantees. Li et al. [180] presented SevDel, combining SGX and blockchain for deletion auditing. However, it focuses solely on the deletion phase and does not prevent local plaintext exposure or address co-ownership and cross-provider consistency. In contrast, our framework holistically supports the entire secure data lifecycle—from creation and sharing to processing without local copies, updating, and verifiable zero-residual deletion across providers.

5.9. CONCLUSION

We propose a provable co-owned data deletion framework for multi-cloud environments, ensuring data remains within the cloud. HSMs manage key lifecycles, while Secure Enclaves enable on-cloud processing without local copies. BMHT enforces strict storage constraints, and GMF ensures global proof verification. Our system provides cryptographic proofs for storage integrity, membership, non-membership, and overwriting, ensuring verifiable deletion. A *zero-residual overwriting* mechanism prevents data recovery. We validate correctness and security through formal analysis and deploy the system on AWS using CloudHSM, Nitro Enclaves, and a Python-based BMHT, evaluating scalability, computational overhead, proof efficiency, and cost analysis.

6

DISCUSSION

Digital forgetting is essential to privacy in the modern digital world, shaping how individuals and organizations control their personal and sensitive data. In today's digital age, where every interaction—be it a social media update, an email, or a shared document—contributes to a growing online footprint, the ability to securely and permanently erase information once it is no longer needed has become an essential pillar of privacy and responsible data stewardship.

The persistence of digital data presents both societal and ethical challenges. On a personal level, people may struggle to remove past mistakes, outdated information, or sensitive details that they no longer wish to be accessible. Without reliable deletion mechanisms, individuals risk having their personal histories permanently available for scrutiny, affecting everything from job prospects to personal relationships. On a larger scale, the inability to erase data raises concerns about unauthorized surveillance, corporate misuse of personal records, and the erosion of digital privacy [215–217].

Recognizing these risks, the General Data Protection Regulation (GDPR) introduced the Right to Be Forgotten (RTBF), formally defined in Article 17 and further elaborated in Recitals 65 and 66 [33, 109, 110], giving individuals legal grounds to request data removal. However, enforcing digital forgetting is far from simple. Traditional deletion methods often fail to guarantee that data is truly gone, especially in distributed or cloud-based environments where copies of information may persist indefinitely.

This thesis tackled the pressing challenge of secure digital forgetting by introducing cryptographic and policy-driven frameworks to enforce controlled and verifiable data expiration. We presented key decay as a cryptographic mechanism to ensure irreversible key expiration, mitigating retrospective adversaries while eliminating reliance on a single entity. To enable flexible audience-specific expiration, we developed the Disjunctive Multi-Level Forgetting Scheme, which allows

selective and staged data deletion, balancing security with usability in distributed environments. Additionally, we introduced a Policy-Based Digital Forgetting framework for co-owned data, extending the decay notion with conjunctive voting to ensure fair, policy-compliant deletion in collaborative digital environments. Finally, we addressed the challenge of verifiable deletion in multi-cloud infrastructures, leveraging cryptographic proofs and hardware security modules to guarantee transparency, integrity, and complete data removal.

Beyond the technical contributions, this thesis advances societal trust and ethical responsibility in data governance. By enforcing verifiable digital forgetting, our solutions empower individuals with greater autonomy over their personal data, enable fair and transparent co-ownership management, and ensure regulatory compliance in cloud-based services and healthcare. These advancements strengthen privacy protections and reinforce the right to be forgotten, establishing a foundation for secure, privacy-preserving, and ethically responsible digital ecosystems.

This chapter presents our findings, societal and ethical implications, limitations, and potential future work.

6.1. RETROSPECTIVE PRIVACY

Ensuring encryption keys expire securely in the digital age is vital for data privacy. Traditional key management relies on explicit deletion or centralized revocation, yet these methods are vulnerable in distributed environments where keys can be archived and exploited. Retrospective adversaries can retrieve and store ephemeral keys, allowing unauthorized decryption long after the data was meant to be inaccessible.

A major weakness of previous approaches is their reliance on ephemeral storage for key expiration. Even with DNS-based expiration, cached keys can persist, making them accessible to adversaries or service providers. The challenge is not just deleting keys but ensuring they become irreversibly useless, even if previously accessed.

This section explores the following research question:

> **RQ1:** How can ephemeral keys be protected against retrospective adversaries?

In Chapter 2, we introduced *Key Decay*, a cryptographic technique ensuring that encryption keys degrade beyond recovery. Unlike conventional approaches that depend on external ephemeral storage, key decay embeds expiration directly into the key material, making future reconstruction computationally infeasible once the number of valid shares drops below the threshold K. In practice, this means that

if an attacker only had access to the share-generating sources (but had not yet reconstructed the key) during the period when the encrypted data was stored in the cloud, those shares become useless after decay.

Key decay operates through entropy-driven passive decay and polynomial-based corruption. Randomness sources continuously alter key structures, while Lagrange interpolation ensures that reconstructing the key becomes impossible as components degrade. Our security analysis confirmed that recovering a decayed key requires infeasible computational resources after surpassing a threshold.

6.2. AUDIENCE-BASED EXPIRATION

Digital expiration should not follow a rigid, one-size-fits-all approach. Different users require varying validity durations; a uniform expiration model fails to accommodate these needs. Some data must remain accessible to specific groups while expiring for others, and users should be able to revoke or extend access dynamically. Traditional key expiration mechanisms enforce a fixed schedule for all users or leave data accessible beyond its intended lifespan, creating security risks.

This section explores the following research question:

> **RQ2:** How can audience-based expiration be achieved across heterogeneous groups with disjunctive access conditions?

In Chapter 3, we introduced the *Disjunctive Multi-Level Forgetting Scheme*, a framework that enables expiration policies tailored to different user groups. Instead of enforcing a single deletion time, our model allows for selective and staged expiration based on user-defined rules. By integrating cryptographic key decay with smart contract enforcement, data access is revoked in a controlled manner without relying on a central authority. Security analysis confirmed that this model prevents unauthorized retention while supporting manual modifications, allowing users to revoke access or extend expiration as needed.

6.3. DATA CO-OWNERSHIP DIGITAL FORGETTING

Co-owned data presents unique access control and deletion challenges in collaborative digital environments. Unlike individually owned files, where deletion is a unilateral decision, co-owned data requires a governance model that allows all stakeholders to participate in the decision-making process. Traditional deletion frameworks either apply uniform expiration policies or lack mechanisms to involve co-owners, making them incompatible with privacy regulations such as the GDPR's Right to be Forgotten (RTBF).

This section explores the following research question:

RQ3: How can we correctly ensure digital data forgetting for co-owned data?

In Chapter 4, we introduced the *Policy-Based Conjunctive Scheme* (PBCS). This framework enforces digital forgetting in shared data by integrating cryptographic access control, voting-based deletion, and policy-driven key decay. Unlike conventional systems, which allow a single owner to control deletion, PBCS ensures that co-owners participate in data expiration through a conjunctive voting mechanism. This guarantees that no single entity can unilaterally delete or retain data without consensus, improving compliance with regulatory requirements. PBCS introduces several key contributions:

- A cloud-based framework that enables secure digital forgetting in co-owned data environments, ensuring compliance with GDPR's Right to be Forgotten (RTBF).
- A governance model where co-owners collectively decide on data deletion using a conjunctive voting mechanism, supporting majority voting and veto-based policies.
- A dynamic access control system that allows revocation and expiration of data access for mutual co-owners while preserving access for others.
- An optimized key decay mechanism that balances passive entropy-driven expiration with active policy enforcement, ensuring controlled and irreversible cryptographic key degradation over time.
- A security and performance evaluation demonstrating the efficiency and feasibility of PBCS in real-world distributed environments.

6.4. VERIFIABLE CO-OWNED DATA DELETION IN MULTI-CLOUD ENVIRONMENTS

Ensuring secure and provable data deletion in multi-cloud environments presents significant challenges due to the lack of transparency in cloud providers' handling of stored information. Traditional deletion models function as a black-box process, offering users only a binary confirmation—either "yes, deleted" or "no, retained"—without verifying that data has been removed. This lack of accountability is especially problematic for co-owned data, where multiple stakeholders must trust that deletion policies are enforced correctly, without hidden backups or unauthorized retention. Additionally, the distributed nature of

multi-cloud environments complicates enforcement, as data can exist in multiple jurisdictions and under varying retention policies.

This section explores the following research question:

> **RQ4:** How can secure deletion of co-owned data be provably verified across multi-cloud infrastructures?

In Chapter 5, we introduced a framework that ensures *verifiable and enforceable deletion across multiple Byzantine cloud providers*, addressing the challenges of transparency and compliance. Unlike traditional deletion mechanisms that rely on service providers' assurances, our model integrates cryptographic proofs to verify that deleted content is irrecoverable while ensuring that sensitive data never leaves the cloud.

The proposed solution employs a multi-layer verification approach. BMHT provides integrity checks for deletion events at the cloud level, while a GMF aggregates deletion proofs across providers, ensuring a tamper-resistant and auditable deletion record. HSMs facilitate the controlled destruction of cryptographic keys, ensuring that decryption is permanently impossible even if remnants of encrypted data persist. Secure enclaves enhance privacy by executing deletion processes within isolated environments, eliminating the risk of external access or data leakage.

To prevent any recoverable traces of deleted data, the framework incorporates zero-residual overwriting based on *NIST* guidelines, ensuring that deleted storage locations are systematically overwritten, preventing forensic recovery. This method guarantees that data removal is both cryptographically irreversible and physically unrecoverable.

6.5. SOCIETAL AND ETHICAL IMPLICATIONS OF DIGITAL FORGETTING

In an increasingly interconnected digital world, the issues of data privacy and the right to be forgotten have become pivotal societal and ethical concerns. This thesis explored these critical issues by providing robust frameworks for secure, verifiable, and policy-driven data deletion across various sectors. Below, we examine three specific scenarios, highlighting how each contributes to societal trust and ethical responsibility through effective digital forgetting mechanisms.

6.5.1. USE CASE 1: CLOUD-BASED SERVICES AND AUTONOMY

Cloud storage solutions offer unparalleled convenience yet frequently compromise users' control over personal data. Ethical challenges emerge when individuals lose the right to manage their digital presence,

often leaving sensitive or outdated information permanently accessible. This undermines both privacy and the fundamental right to be forgotten.

Chapters 2 & 3 address this ethical and societal dilemma by introducing cryptographic key decay and audience-based expiration frameworks. These solutions enable users and groups to dynamically control data retention and deletion, aligning closely with personal privacy preferences and reinforcing digital autonomy. Societally, this builds trust in cloud-based services, reassuring users that their data privacy rights are respected and upheld.

6.5.2. USE CASE 2: COLLABORATIVE DIGITAL AUTHORSHIP AND FAIRNESS

Managing digital content among multiple authors can be ethically complex, especially when deletion and modification rights are ambiguous or unfairly applied. This scenario is common in multi-author research projects, where contributors have varied access permissions and shared responsibilities. Ethical and societal issues arise when outdated drafts or sensitive data are not effectively deleted, potentially violating authors' rights and undermining collaborative trust.

Chapter 4 provides a policy-based collaborative governance framework for multi-author scenarios. It allows project participants to dynamically manage copyright terms and enforce data deletion policies collaboratively. By securely erasing outdated drafts and clearly defining access and modification rights, this approach ensures compliance with the right to be forgotten and fosters fairness among co-authors. Societally, this strengthens trust and transparency within collaborative environments, facilitating respectful and responsible intellectual contributions.

6.5.3. USE CASE 3: HEALTHCARE DATA MANAGEMENT AND PATIENT PRIVACY

Healthcare institutions handle highly sensitive personal data governed by strict laws like GDPR and European Health Data Space (EHDS). Ethical challenges arise when patient data is retained beyond necessity or inadequately managed, undermining patient trust and the fundamental right to privacy.

Chapter 5 addresses these concerns through a cryptographic framework enabling secure, verifiable deletion and collaborative data governance between hospitals and research entities. This solution ensures compliance with strict privacy regulations and reinforces informed consent, empowering patients to trust that their sensitive data will not be misused or retained indefinitely. Concerning society, this solution enhances transparency and trust in healthcare data practices, promoting ethical responsibility and safeguarding patient rights.

Overall, this thesis enhanced societal trust and ethical standards, ensuring that digital forgetting becomes a practical reality across sectors, effectively safeguarding privacy and reinforcing the right to be forgotten (RTBF).

6.6. LIMITATIONS AND FUTURE WORK

This section outlines the key limitations identified throughout the thesis and proposes future research directions. These reflections are grounded in the core contributions presented across the preceding chapters, highlighting current challenges and opportunities for advancing secure, privacy-preserving digital forgetting mechanisms.

6.6.1. KEY DECAY

While the proposed key decay mechanism provides a foundational approach to secure digital forgetting, its initial implementation has several limitations. The expiration mechanism enforced a rigid, fixed decay rate, making it inadequate for scenarios requiring adaptive or user-specific expiration schedules. Furthermore, it lacked manual revocation and extension capabilities, restricting users from dynamically controlling the lifecycle of their keys. The system's reliance on external entropy sources also introduced reliability concerns in constrained environments. Critically, the original design did not support multi-owner data governance, limiting its applicability to collaborative or shared data settings.

To address these limitations, enhancements were introduced across subsequent chapters. In Chapter 3, the *Disjunctive Multi-Level Forgetting Scheme* extended key decay by introducing hierarchical and audience-specific expiration controls, enabling flexible retention policies across different user groups. Manual revocation mechanisms were added to allow dynamic access updates and revocable data sharing. Chapter 4 further refined this approach through the *Policy-Based Active Decay Scheme*, which embedded policy-driven triggers and introduced conjunctive voting to support coordinated key expiration in co-owned data environments. These evolutions demonstrated how foundational key decay could be expanded to address real-world privacy and collaboration requirements.

Future work should focus on strengthening key decay's resilience to emerging threats. This includes integrating quantum-resistant cryptographic primitives to safeguard against future computational advances and refining entropy-sourcing techniques for enhanced robustness in limited-resource scenarios. Reducing the computational complexity of polynomial-based corruption mechanisms will also be crucial for deployment in constrained environments. Lastly, incorporating key decay into

trusted execution environments (TEEs) could provide hardware-level enforcement of irreversible key expiration, closing the gap between cryptographic guarantees and system-level assurances.

6.6.2. DISJUNCTIVE MULTI-LEVEL DIGITAL FORGETTING SCHEME

While audience-specific expiration enhances privacy granularity, practical deployment poses several challenges. One significant limitation is the computational cost associated with smart contract execution. Enforcing access control and expiration policies via blockchain or cryptographic verification imposes considerable overhead, making real-time or frequent policy updates resource-intensive and potentially impractical for large-scale systems.

Moreover, supporting context-aware expiration remains technically complex, where content visibility adapts to evolving life circumstances or social contexts. Users may experience changes in privacy preferences due to job transitions, relationship changes, or shifts in social circles. However, these contextual signals are often implicit and not explicitly expressed or encoded in system rules. Designing cryptographic erasure mechanisms that automatically adapt to such nuanced personal dynamics is particularly difficult, especially when data is published anonymously or without links to other contextual data sources. Although service providers could, in theory, aggregate auxiliary metadata to infer user context, this raises further privacy concerns and conflicts with data minimization principles.

Future Work should explore reducing the reliance on high-cost blockchain operations through lightweight verification techniques or off-chain policy enforcement models. Incorporating user-driven heuristics, such as access frequency, social proximity, or data aging, could enable dynamic and context-sensitive expiration policies. Future systems should balance cryptographic rigidity with flexible policy adaptation, potentially leveraging privacy-preserving context modeling to reflect real-world privacy needs without compromising user anonymity or increasing metadata exposure.

6.6.3. CONJUNCTIVE SCHEME FOR DIGITAL FORGETTING OF CO-OWNED DATA

One of the primary challenges in PBCS lies in coordinating decay and share refresh operations across highly distributed systems. Reliable key expiration requires precise synchronization; otherwise, premature key corruption may occur, jeopardizing data availability before expiration. Additionally, managing overlapping policies among multiple owners with distinct access and deletion rights necessitates more sophisticated revocation mechanisms to avoid policy conflicts. From a security

perspective, hardware-accelerated attacks, such as those leveraging FPGAs, present an emerging threat. For instance, studies have shown that FPGA-based acceleration can reduce RSA decryption time by up to 4×. Although PBCS relies on sequential share aggregation, which inherently resists parallel brute-forcing, further analysis is needed to assess its robustness under advanced hardware-assisted attack models.

Future research should explore the integration of Homomorphic Encryption to support provable data deletion without requiring complete trust in the cloud provider. Another promising direction is the adoption of quantum-resistant cryptographic primitives to ensure resilience against future quantum computing threats. Furthermore, extending PBCS to support multi-dimensional decay strategies—where multiple attributes (e.g., time, access frequency, or sensitivity) drive decay in parallel—could offer more granular and context-aware control over collaborative data expiration.

6.6.4. PROVABLE DELETION OF CO-OWNED DATA IN CLOUD ENVIRONMENT

One key challenge in this approach is ensuring deletion consistency across multi-cloud infrastructures, where different cloud providers operate under distinct policies and data replication mechanisms. While cryptographic proofs confirm deletion in primary storage locations, ensuring the complete removal of all redundant copies, backup instances, and distributed storage nodes remains an open issue.

Future research will focus on developing zero-knowledge proof-based public deletion verification, allowing users to verify deletion across multiple cloud providers without exposing sensitive metadata or relying on provider trust. This will enable a privacy-preserving, publicly auditable deletion mechanism that ensures compliance while maintaining confidentiality. Additionally, refining deletion tracking mechanisms to account for the intricacies of multi-cloud storage, including data fragmentation, replication policies, and cross-jurisdictional constraints, will further strengthen the framework's ability to provide complete and verifiable digital forgetting.

6.6.5. TOWARDS MACHINE UNLEARNING IN PRIVACY-PRESERVING SYSTEMS

While this thesis addresses secure digital forgetting through cryptographic key decay, audience/policy-based expiration, and multi-cloud verifiability, future work must also consider the emerging challenge of machine unlearning. As artificial intelligence systems increasingly rely on user data, it becomes essential to ensure that deleted information does not persist in trained models. Traditional deletion guarantees are

no longer sufficient to comply with GDPR [218–221].

One key direction involves certified model update mechanisms, where training procedures eliminate the influence of deleted data without requiring complete retraining. These methods would ensure that removed data points no longer affect model predictions, offering stronger privacy guarantees.

Another extension concerns verifiable machine unlearning. Building on the verifiability principles of this thesis, such as multi-party deletion proofs and auditable deletion tracking, future work should aim to develop similar guarantees for model behavior. This would enable users to verify that their data has been removed from storage and inference processes.

Policy-aware machine unlearning offers another natural evolution. The policy-driven mechanisms developed in this thesis, including expiration, revocation, and voting-based control, can serve as automated triggers for model unlearning. This would allow learning systems to adjust by dynamic access rights or regulatory compliance rules.

In collaborative environments, multi-owner or co-owned data unlearning becomes essential. Extending the PBCS, future work could explore collaborative unlearning for shared data. This includes defining ownership-aware model updates where all stakeholders must agree on the unlearning scope and ensuring equitable revocation of influence in models trained on jointly contributed data.

Lastly, the entropy-based key decay mechanisms introduced in this work suggest a novel approach: entropy-driven model decay. Future systems could gradually diminish a model's reliance on expired or outdated data inputs, supporting a more natural and continuous forgetting process without requiring manual retraining.

Together, these directions expand the scope of digital forgetting from secure deletion to machine unlearning, laying the groundwork for privacy-preserving AI systems that respect data ownership and the right to be forgotten (RTBF).

REFERENCES

[1] Statista. *Volume of Data Created, Consumed, and Stored Worldwide*. Accessed: 2025-03-15. url: https://www.statista.com/statistics/871513/worldwide-data-created/.

[2] T. Coughlin. “175 Zettabytes by 2025”. In: *Forbes* (2018). Accessed: 2025-03-16. url: https://www.forbes.com/sites/tomcoughlin/2018/11/27/175-zettabytes-by-2025/.

[3] M. A. Darwish, E. Yafi, A. H. Almasri, and M. F. Zuhairi. “Privacy and security of cloud computing: a comprehensive review of techniques and challenges”. In: *International Journal of Engineering Trends and Technology* 7.4.29 (2018), pp. 239–246.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, *et al.* “A view of cloud computing”. In: *Communications of the ACM* 53.4 (2010), pp. 50–58.

[5] Q. Zhang, L. Cheng, and R. Boutaba. “Cloud computing: state-of-the-art and research challenges”. In: *Journal of internet services and applications* 1 (2010), pp. 7–18.

[6] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility”. In: *Future Generation computer systems* 25.6 (2009), pp. 599–616.

[7] S. ACHAR, H. PATEL, and S. HUSSAIN. “Data security in cloud: A review”. In: *Asian Journal of Advances in Research* 17.4 (2022), pp. 76–83.

[8] Edge Delta. *How Many Companies Use Cloud Computing in 2024?* Accessed: 2025-03-15. url: https://edgedelta.com/company/blog/how-many-companies-use-cloud-computing-in-2024.

[9] J. M. Wing. “The data life cycle”. In: *Harvard Data Science Review* 1.1 (2019), p. 6.

[10] B. Li, Y. Fu, and K. Wang. “A Review on Cloud Data Assured Deletion”. In: *2022 Global Conference on Robotics, Artificial Intelligence and Information Technology (GCRAIT)*. IEEE. 2022, pp. 451–457.

[11] S. Diesburg, C. Meyers, M. Stanovich, A.-I. A. Wang, and G. Kuenning. “Trueerase: Leveraging an auxiliary data path for per-file secure deletion”. In: *ACM Transactions on Storage (TOS)* 12.4 (2016), pp. 1–37.

[12] J. Xiong, F. Li, J. Ma, X. Liu, Z. Yao, and P. S. Chen. “A full lifecycle privacy protection scheme for sensitive data in cloud computing”. In: *Peer-to-peer Networking and Applications* 8 (2015), pp. 1025–1037.

[13] K. M. Ramokapane, J. Such, and A. Rashid. “What users want from cloud deletion and the information they need: A participatory action study”. In: *ACM Transactions on Privacy and Security* 26.1 (2022), pp. 1–34.

[14] J. Reardon, D. Basin, and S. Capkun. “On secure data deletion”. In: *IEEE security & privacy* 12.3 (2014), pp. 37–44.

[15] A. Greenberg. *This machine kills secrets: Julian Assange, the Cypherpunks, and their fight to empower whistleblowers*. Penguin, 2013.

[16] G. S. Yılmaz, F. Gasaway, B. Ur, and M. Mondal. “Perceptions of retrospective edits, changes, and deletion on social media”. In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 15. 2021, pp. 841–852.

[17] O. I. Abiodun, M. Alawida, A. E. Omolara, and A. Alabdulatif. “Data provenance for cloud forensic investigations, security, challenges, solutions and future perspectives: A survey”. In: *Journal of King Saud University-Computer and Information Sciences* 34.10 (2022), pp. 10217–10245.

[18] Dropbox Community. *Deleted folder reappeared after a couple of years*. Accessed: 2023-04-21. 2017. url: https://www.dropboxforum.com/discussions/101001013/re-deleted-folder-re-appeared-after-a-couple-of-years/202656/replies/203008.

[19] Cimpanu, Catalin. *Dropbox Kept Files Around for Years Due to ’Delete’ Bug*. Accessed: 2023-04-21. 2017. url: https://www.bleepingcomputer.com/news/software/dropbox-kept-files-around-for-years-due-to-delete-bug/.

[20] B.-J. Koops. “Forgetting footprints, shunning shadows: A critical analysis of the right to be forgotten in big data practice”. In: *SCRIPTed* 8 (2011), p. 229.

[21] European Data Protection Supervisor. *Glossary - European Data Protection Supervisor*. https://www.edps.europa.eu/data-protection/data-protection/glossary/d_en. Accessed: April 14, 2025. 2025.

[22] S. Shastri, M. Wasserman, and V. Chidambaram. “GDPR anti-patterns: How design and operation of modern cloud-scale systems conflict with GDPR”. In: *arXiv preprint arXiv:1911.00498* (2019).

[23] eWEEK Editors. *The Dangers of Obsolete and Redundant Data*. Accessed: 2025-03-15. url: https://www.eweek.com/security/rot-data-dark-data/.

[24] SecurityWeek Staff. *Solving the Quantum Decryption ‘Harvest Now, Decrypt Later’ Problem*. https://www.securityweek.com/solving-quantum-decryption-harvest-now-decrypt-later-problem/. Accessed: April 2025. 2023.

[25] F. Thouvenin, P. Hettich, H. Burkert, and U. Gasser. *Remembering and Forgetting in the Digital Age*. Vol. 38. Springer, 2018.

[26] D. Sisto. *Remember me: Memory and forgetting in the digital age*. John Wiley & Sons, 2021.

[27] A. Schlesinger, E. Chandrasekharan, C. A. Masden, A. S. Bruckman, W. K. Edwards, and R. E. Grinter. “Situated anonymity: Impacts of anonymity, ephemerality, and hyper-locality on social media”. In: *Proceedings of the 2017 CHI conference on human factors in computing systems*. 2017, pp. 6912–6924.

[28] T. Schnitzler, C. Utz, F. M. Farke, C. Pöpper, and M. Dürmuth. “User perception and expectations on deleting instant messages—or—“what happens if i press this button?”” In: *Regulation (GDPR)* 4.5 (2018).

[29] Y. Tang, P. P. Lee, J. C. Lui, and R. Perlman. “Secure overlay cloud storage with access control and assured deletion”. In: *IEEE Transactions on dependable and secure computing* 9.6 (2012), pp. 903–916.

[30] M. Bishop, E. R. Butler, K. Butler, C. Gates, and S. Greenspan. “Forgive and forget: return to obscurity”. In: *Proceedings of the 2013 New Security Paradigms Workshop*. USA: ACM, 2013, pp. 1–10.

[31] M. L. Ambrose, N. Friess, and J. Van Matre. “Seeking digital redemption: The future of forgiveness in the Internet age”. In: *Santa Clara Computer & High Tech. LJ* 29 (2012), p. 99.

[32] J. Rosen. “The web means the end of forgetting”. In: *The New York Times* 21 (2010).

[33] GDPR.eu. *Everything You Need to Know About the "Right to Be Forgotten"*. Accessed: 2025-03-15. url: https://gdpr.eu/right-to-be-forgotten/.

[34] P. Voigt and A. Von dem Bussche. “The eu general data protection regulation (gdpr)”. In: *A practical guide, 1st ed., Cham: Springer International Publishing* 10.3152676 (2017), pp. 10–5555.

[35] L. de la Torre. “A guide to the california consumer privacy act of 2018”. In: *Available at SSRN 3275571* (2018).

[36] J. Hughes, M. E. Kaminski, J. Snow, and F. T. Wu. “Symposium: The california consumer privacy act”. In: *Loyola of Los Angeles Law Review* 54 (2021).

[37] European Commission. *Common European Data Spaces*. 2025. url: https://digital-strategy.ec.europa.eu/en/policies/data-spaces.

[38] European Commission. *European Health Data Space*. https://health.ec.europa.eu/ehealth-digital-health-and-care/european-health-data-space_en. Accessed: 2025-03-15.

[39] Federal Register. “Modifications to the HIPAA privacy, security, enforcement, and breach notification rules under the Health Information Technology for Economic and Clinical Health Act and the Genetic Information Nondiscrimination Act; other modifications to the HIPAA Rules”. In: *Other Modifications to the HIPAA Rules* 78.17 (2013), pp. 5566–5702.

[40] C. S. Redhead. *HIPAA Privacy, Security, Enforcement, and Breach Notification Standards*. 2015.

[41] N. Allahrakha. “Balancing cyber-security and privacy: legal and ethical considerations in the digital age”. In: *Legal Issues in the digital Age* 2 (2023), pp. 78–121.

[42] N. Azam, L. Michala, S. Ansari, and N. B. Truong. “Data privacy threat modelling for autonomous systems: a survey from the gdpr’s perspective”. In: *IEEE Transactions on Big Data* 9.2 (2022), pp. 388–414.

[43] T. Schnitzler, S. Mirza, M. Dürmuth, and C. Pöpper. “Sok: Managing longitudinal privacy of publicly shared personal online data”. In: *Proceedings on Privacy Enhancing Technologies* (2021).

[44] C. Castelluccia, E. De Cristofaro, A. Francillon, and M.-A. Kaafar. “Ephpub: Toward robust ephemeral publishing”. In: *2011 19th IEEE International Conference on Network Protocols*. Canada: IEEE, 2011, pp. 165–175.

[45] G. Amjad, M. S. Mirza, and C. Pöpper. “Forgetting with puzzles: using cryptographic puzzles to support digital forgetting”. In: *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*. USA: ACM, 2018, pp. 342–353.

[46] R. E. Mohamed and S. Chiasson. “Online privacy and aging of digital artifacts”. In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. 2018, pp. 177–195.

[47] Y. Gurevich, E. Hudis, and J. M. Wing. “Inverse privacy”. In: *Communications of the ACM* 59.7 (2016), pp. 38–42.

[48] C. Hu, Y. Xu, P. Liu, J. Yu, S. Guo, and M. Zhao. “Enabling cloud storage auditing with key-exposure resilience under continual key-leakage”. In: *Information Sciences* 520 (2020), pp. 15–30.

[49] W. J. Blanke. “Data loss prevention using an ephemeral key”. In: *2011 International Conference on High Performance Computing & Simulation*. IEEE. 2011, pp. 412–418.

[50] B. Greschbach, G. Kreitz, and S. Buchegger. “The devil is in the metadata—new privacy challenges in decentralised online social networks”. In: *2012 IEEE international conference on pervasive computing and communications workshops*. IEEE. 2012, pp. 333–339.

[51] A. Besmer and H. Richter Lipford. “Moving beyond untagging: photo privacy in a tagged world”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 1563–1572.

[52] W. Yi, C. Wang, J. Chen, S. Kuzmin, I. Gerasimov, and X. Cheng. “DSDM-TCSE: Deterministic storage and deletion mechanism for trusted cloud service environments”. In: *Future Generation Computer Systems* 165 (2025), p. 107611.

[53] F. Hao, D. Clarke, and A. F. Zorzo. “Deleting secret data with public verifiability”. In: *IEEE Transactions on Dependable and Secure Computing* 13.6 (2015), pp. 617–629.

[54] C. Yang, Y. Liu, X. Tao, and F. Zhao. “Publicly verifiable and efficient fine-grained data deletion scheme in cloud computing”. In: *IEEE Access* 8 (2020), pp. 99393–99403.

[55] A. Goldsteen, T. Douek, Y. Cohen, I. Gokhman, O. Keren-Ackerman, G. Katsovich, G. Weintraub, and D. Ben-Ari. “Forgotten@ Scale: A Practical Solution for Implementing the Right To Be Forgotten in Large-Scale Systems”. In: *arXiv preprint arXiv:1910.13784* (2019).

[56] M. A. Darwish and A. Zarras. “Digital Forgetting Using Key Decay”. In: *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing*. 2023, pp. 34–41.

[57] IBM. *Netezza and IBM Cloud Pak for Data: A knockout combo for tough data*. https://www.ibm.com/blogs/journey-to-ai/2020/06/netezza-and-ibm-cloud-pak-a-knockout-combo-for-tough-data/. 2020.

[58] Cisco. *Cisco Annual Internet Report (2018–2023) White Paper*. https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html. 2020.

[59] V. Mayer-Schönberger. *Delete: The virtue of forgetting in the digital age*. USA: Princeton University Press, 2011.

[60] E. Politou, E. Alepis, and C. Patsakis. “Forgetting personal data and revoking consent under the GDPR: Challenges and proposed solutions”. In: *Journal of cybersecurity* 4.1 (2018).

[61] E. Shein. “Ephemeral data”. In: *Communications of the ACM* 56.9 (2013), pp. 20–22.

[62] G. Wegberg, H. Ritzdorf, and S. Capkun. *Multi-User Secure Deletion on Agnostic Cloud Storage*. Tech. rep. ETH Zurich, 2017.

[63] E. F. Villaronga, P. Kieseberg, and T. Li. “Humans forget, machines remember: Artificial intelligence and the right to be forgotten”. In: *Computer Law & Security Review* 34.2 (2018), pp. 304–313.

[64] R. Perlman. *The ephemerizer: Making data disappear*. 2005.

[65] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy. “Vanish: Increasing Data Privacy with Self-Destructing Data.” In: *USENIX security symposium*. Vol. 316. USA: USENIX, 2009, pp. 10–5555.

[66] A. Zarras, K. Kohls, M. Dürmuth, and C. Pöpper. “Neuralyzer: Flexible expiration times for the revocation of online data”. In: *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*. USA: ACM, 2016, pp. 14–25.

[67] S. Reimann and M. Dürmuth. “Timed revocation of user data: long expiration times from existing infrastructure”. In: *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. USA: ACM, 2012, pp. 65–74.

[68] Verisign. *Domain Names Industry Brief*. https://www.verisign.com/en_US/domain-names/dnib/index.xhtml. 2020.

[69] B. Das and D. Chakrabarty. “Lagrange’s interpolation formula: representation of numerical data by a polynomial curve”. In: *International Journal of Mathematics Trend and Technology* 34.2 (2016), pp. 23–31.

[70] A. Shamir. “How to share a secret”. In: *Communications of the ACM* 22.11 (1979), pp. 612–613.

[71] J. Daemen. *The Design of Rijndael:The Advanced Encryption Standard (AES) (Information Security and Cryptography)*. USA: Springer; 2nd ed. 2020 edition, 2020.

[72] The Chain Bulletin. *The Chain Bulletin*. https://chainbulletin.com/proof-of-work-explained-in-simple-terms. 2019.

[73] *Order of user liked tweets - X API v2 - X Developers*. Accessed: 2025-03-23. url: https://devcommunity.x.com/t/order-of-user-liked-tweets/164227.

[74] *Pagination - X - API*. Accessed: 2025-03-23. url: https://docs.x.com/x-api/fundamentals/pagination.

[75] *Implementation: Pagination | YouTube Data API*. Accessed: 2025-03-23. url: https://developers.google.com/youtube/v3/guides/implementation/pagination.

[76] L. Gupta. *REST API Design: Filtering, Sorting, and Pagination*. https://www.moesif.com/blog/technical/api-design/REST-API-Design-Filtering-Sorting-and-Pagination/. Accessed: 2025-03-23.

[77] A. A. Rodríguez, L. B. Bruno, and F. Rapetti. "Whitney edge elements and the Runge phenomenon". In: *Journal of Computational and Applied Mathematics* 427 (2023), p. 115117.

[78] J.-P. Berrut and L. N. Trefethen. "Barycentric lagrange interpolation". In: *SIAM review* 46.3 (2004), pp. 501–517.

[79] N. J. Higham. "The numerical stability of barycentric Lagrange interpolation". In: *IMA Journal of Numerical Analysis* 24.4 (2004), pp. 547–556.

[80] O. Harrison and J. Waldron. "Efficient acceleration of asymmetric cryptography on graphics hardware". In: *International conference on cryptology in africa*. Berlin: Springer, 2009, pp. 350–367.

[81] T. Schnitzler, M. Dürmuth, and C. Pöpper. "Towards contractual agreements for revocation of online data". In: *IFIP International Conference on ICT Systems Security and Privacy Protection*. Lisbon: Springer, 2019, pp. 374–387.

[82] C. Yang, Y. Liu, F. Zhao, and S. Zhang. "Provable data deletion from efficient data integrity auditing and insertion in cloud storage". In: *Computer Standards & Interfaces* 82 (2022), p. 103629.

[83] J. Xiong, L. Chen, M. Z. A. Bhuiyan, C. Cao, M. Wang, E. Luo, and X. Liu. "A secure data deletion scheme for IoT devices through key derivation encryption and data analysis". In: *Future Generation Computer Systems* 111 (2020), pp. 741–753.

[84] A. Ginart, M. Guan, G. Valiant, and J. Y. Zou. "Making AI forget you: Data deletion in machine learning". In: *Advances in Neural Information Processing Systems* 32.1 (2019).

[85] M. Minaei, M. Mondal, P. Loiseau, K. Gummadi, and A. Kate. “Lethe: Conceal content deletion from persistent observers”. In: *Proceedings on Privacy Enhancing Technologies* 2019.1 (2019), pp. 206–226.

[86] A. Abouzied and J. Chen. “Harnessing data loss with forgetful data structures”. In: *Proceedings of the Sixth ACM Symposium on Cloud Computing*. USA: ACM, 2015, pp. 168–173.

[87] M. A. Darwish and G. Smaragdakis. “Disjunctive Multi-Level Digital Forgetting Scheme”. In: *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 2024, pp. 112–121.

[88] F. Armknecht, J.-M. Bohli, G. O. Karame, and F. Youssef. “Transparent data deduplication in the cloud”. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 2015, pp. 886–900.

[89] K. Ren, C. Wang, and Q. Wang. “Security challenges for the public cloud”. In: *IEEE Internet computing* 16.1 (2012), pp. 69–73.

[90] P. V. Mockapetris. *RFC1035: Domain Names-Implementation and Specification*. 1987.

[91] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. “Chord: A scalable peer-to-peer lookup service for internet applications”. In: *ACM SIGCOMM computer communication review* 31.4 (2001), pp. 149–160.

[92] X. Ma, L. Xu, and F. Zhang. “Oblivious transfer with timed-release receiver’s privacy”. In: *Journal of Systems and Software* 84.3 (2011), pp. 460–464.

[93] Q. Liu, G. Wang, and J. Wu. “Time-based proxy re-encryption scheme for secure data sharing in a cloud environment”. In: *Information sciences* 258 (2014), pp. 355–370.

[94] L. Xu, F. Zhang, and S. Tang. “Timed-release oblivious transfer”. In: *Security and Communication Networks* 7.7 (2014), pp. 1138–1149.

[95] Z. Yue, Y. Yao, W. Li, and N. Yu. “Atdd: Fine-grained assured time-sensitive data deletion scheme in cloud storage”. In: *ICC 2022-IEEE International Conference on Communications*. IEEE. 2022, pp. 3448–3453.

[96] J. Hong, K. Xue, Y. Xue, W. Chen, D. S. Wei, N. Yu, and P. Hong. “TAFC: Time and attribute factors combined access control for time-sensitive data in public cloud”. In: *IEEE Transactions on Services Computing* 13.1 (2017), pp. 158–171.

[97] C. Yu, Y. Zhan, and M. Sohail. “SDSM: Secure Data Sharing for Multilevel Partnerships in IoT Based Supply Chain”. In: *Symmetry* 14.12 (2022), p. 2656.

[98] W. Wang, H. Huang, Z. Yin, T. R. Gadekallu, M. Alazab, and C. Su. “Smart contract token-based privacy-preserving access control system for industrial Internet of Things”. In: *Digital Communications and Networks* 9.2 (2023), pp. 337–346.

[99] Y. Shen, B. Yu, S. Lai, X. Yuan, S.-F. Sun, J. K. Liu, and S. Nepal. “OblivSend: Secure and Ephemeral File Sharing Services with Oblivious Expiration Control”. In: *International Conference on Information Security*. Springer. Switzerland: Springer, 2022, pp. 269–289.

[100] F. Karami, D. Basin, and E. B. Johnsen. “DPL: A Language for GDPR Enforcement”. In: *2022 IEEE 35th Computer Security Foundations Symposium (CSF)*. IEEE. Haifa: IEEE, 2022, pp. 112–129.

[101] Datasnaek. *YouTube Trending Video Statistics Dataset*. Accessed: June 29, 2023. url: https://www.kaggle.com/datasets/datasnaek/youtube-new.

[102] R. Sharma. *Cloud Storage Market - Global Industry Analysis*. https://growthmarketreports.com/report/cloud-storage-market-global-industry-analysis. Accessed: April 20, 2024. 2023.

[103] S. Mirza and C. Pöpper. “My Past Dictates my Present: Relevance, Exposure, and Influence of Longitudinal Data on Facebook”. In: *Workshop on Usable Security and Privacy (USEC)*. 2021.

[104] F. Hublet, D. Basin, and S. Krstić. “Enforcing the GDPR”. In: *European Symposium on Research in Computer Security*. Springer. 2023, pp. 400–422.

[105] A. Murillo, A. Kramm, S. Schnorf, and A. De Luca. “"If I press delete, it’s gone"-User Understanding of Online Data Deletion and Expiration”. In: *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*. USA: USENIX, 2018, pp. 329–339.

[106] S. Ahern, D. Eckles, N. S. Good, S. King, M. Naaman, and R. Nair. “Over-exposed? Privacy patterns and considerations in online and mobile photo sharing”. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*. USA: ACM, 2007, pp. 357–366.

[107] M. T. Khan, M. Hyun, C. Kanich, and B. Ur. “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*. 2018, pp. 1–12.

[108] K. M. Ramokapane, A. Rashid, and J. M. Such. “Assured deletion in the cloud: requirements, challenges and future directions”. In: *Proceedings of the 2016 ACM on Cloud Computing Security Workshop*. 2016, pp. 97–108.

[109] European Union. *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)*. 2016. url: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation.

[110] A. Deac *et al.* "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". In: *Perspectives of Law and Public Administration* 7.2 (2018), pp. 151–156.

[111] S. Garg, S. Goldwasser, and P. N. Vasudevan. "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. 2020, pp. 373–402.

[112] T. Bertram, E. Bursztein, S. Caro, H. Chao, R. Chin Feman, P. Fleischer, A. Gustafsson, J. Hemerly, C. Hibbert, L. Invernizzi, *et al.* "Five years of the right to be forgotten". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. 2019, pp. 959–972.

[113] L. Zeng, Z. Shi, S. Xu, and D. Feng. "SafeVanish: An Improved Data Self-Destruction for Protecting Data Privacy". In: *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. IEEE. 2010, pp. 521–528.

[114] F. Nourmohammadzadeh Motlagh, S. A. Alhosseini, F. Cheng, and C. Meinel. "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*. 2023, pp. 264–268.

[115] A. Dhir, P. Kaur, K. Lonka, and M. Nieminen. "Why do adolescents untag photos on Facebook?" In: *Computers in Human Behavior* 55 (2016), pp. 1106–1115.

[116] H. Alshareef, R. Pardo, G. Schneider, and P. Picazo-Sanchez. "A collaborative access control framework for online social networks". In: *Journal of Logical and Algebraic Methods in Programming* 114 (2020), p. 100562.

[117] R. Geambasu, T. Kohno, A. Krishnamurthy, A. Levy, H. M. Levy, P. Gardner, and V. Moscaritolo. "New directions for self-destructing data". In: *University of Washington, Tech. Rep. UW-CSE-11-08-01* (2011).

[118] C. Yang, X. Tao, F. Zhao, and Y. Wang. "Secure data transfer and deletion from counting bloom filter in cloud computing". In: *Chinese Journal of Electronics* 29.2 (2020), pp. 273–280.

[119] A. Cavoukian *et al.* “Privacy by design: The seven foundational principles”. In: *IAPP Resource Center* (2021).

[120] O. Ayalon and E. Toch. “Retrospective privacy: Managing longitudinal privacy in online social networks”. In: *Proceedings of the Ninth Symposium on Usable Privacy and Security*. UK: ACM, 2013, pp. 1–13.

[121] P. Ilia, B. Carminati, E. Ferrari, P. Fragopoulou, and S. Ioannidis. “SAMPAC: Socially-aware collaborative multi-party access control”. In: *proceedings of the seventh ACM on conference on data and application security and privacy*. 2017, pp. 71–82.

[122] B. Carminati and E. Ferrari. “Collaborative access control in on-line social networks”. In: *7th international conference on collaborative computing: Networking, applications and worksharing (CollaborateCom)*. IEEE. 2011, pp. 231–240.

[123] J. M. Such and N. Criado. “Resolving multi-party privacy conflicts in social media”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.7 (2016), pp. 1851–1863.

[124] W. Susilo, P. Jiang, J. Lai, F. Guo, G. Yang, and R. H. Deng. “Sanitizable access control system for secure cloud storage against malicious data publishers”. In: *IEEE Transactions on Dependable and Secure Computing* 19.3 (2021), pp. 2138–2148.

[125] L. Zhou, A. Fu, Y. Mu, H. Wang, S. Yu, and Y. Sun. “Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system”. In: *Information Sciences* 545 (2021), pp. 254–276.

[126] J. Bethencourt, A. Sahai, and B. Waters. “Ciphertext-policy Attribute-based Encryption”. In: *2007 IEEE symposium on security and privacy (SP’07)*. IEEE. 2007, pp. 321–334.

[127] E. De Cristofaro, C. Soriente, G. Tsudik, and A. Williams. “Hummingbird: Privacy at the time of twitter”. In: *2012 IEEE Symposium on security and privacy*. IEEE. 2012, pp. 285–299.

[128] K. M. Ramokapane, A. Rashid, and J. M. Such. ““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)*. 2017, pp. 241–256.

[129] J. Li, R. Zhang, Y. Lu, J. Han, Y. Zhang, W. Zhang, and X. Dong. “Multiauthority attribute-based encryption for assuring data deletion”. In: *IEEE Systems Journal* 17.2 (2022), pp. 2029–2038.

[130] K. P. Coopamootoo and T. Groß. “Why privacy is all but forgotten”. In: *Proceedings on Privacy Enhancing Technologies* (2017).

[131] M. Mondal, G. S. Yilmaz, N. Hirsch, M. T. Khan, M. Tang, C. Tran, C. Kanich, B. Ur, and E. Zheleva. “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*. 2019, pp. 991–1008.

[132] S.-C. Yeh, M.-Y. Su, H.-H. Chen, and C.-Y. Lin. “An efficient and secure approach for a cloud collaborative editing”. In: *Journal of Network and Computer Applications* 36.6 (2013), pp. 1632–1641.

[133] M. Bellare and P. Rogaway. “Introduction to modern cryptography”. In: *Lecture Notes* (2001).

[134] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. Wei, and P. Hong. “An Attribute-based Controlled Collaborative Access Control Scheme for Public Cloud Storage”. In: *IEEE Transactions on Information Forensics and Security* 14.11 (2019), pp. 2927–2942.

[135] S. Arora and P. K. Atrey. “Secure collaborative editing using secret sharing”. In: *2021 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE. USA: IEEE, 2021, pp. 1–6.

[136] D. Quadling. “Lagrange’s interpolation formula”. In: *The Mathematical Gazette* 50.374 (1966), pp. 372–375.

[137] F. Ortega, R. Lara-Cabrera, Á. González-Prieto, and J. Bobadilla. “Providing reliability in recommender systems through Bernoulli matrix factorization”. In: *Information sciences* 553 (2021), pp. 110–128.

[138] M. Berz and G. Hoffstätter. “Computation and application of Taylor polynomials with interval remainder bounds”. In: *Reliable Computing* 4.1 (1998), pp. 83–97.

[139] D. Kozen and M. Timme. “Indefinite summation and the Kronecker delta”. In: (2007).

[140] P. Cibik, P. Dobias, S. Ricci, J. Hajny, L. Malina, P. Jedlicka, and D. Smekal. “Pushing AES-256-GCM to Limits: Design, Implementation and Real FPGA Tests”. In: *International Conference on Applied Cryptography and Network Security*. Springer. 2024, pp. 303–318.

[141] R. A. Mollin. *RSA and public-key cryptography*. Chapman and Hall/CRC, 2002.

[142] N. Cao, A. O’Neill, and M. Zaheri. “Toward RSA-OAEP without random oracles”. In: *IACR International Conference on Public-Key Cryptography*. Springer. 2020, pp. 279–308.

[143] H. Gilbert and H. Handschuh. “Security analysis of SHA-256 and sisters”. In: *International workshop on selected areas in cryptography*. Springer. 2003, pp. 175–193.

[144] C.-C. Chang, N.-T. Huynh, and H.-D. Le. “Lossless and unlimited multi-image sharing based on Chinese remainder theorem and Lagrange interpolation”. In: *Signal processing* 99 (2014), pp. 159–170.

[145] Z. Zhang, Q. Wang, and Z. Zhang. “Harmonic vector error analysis based on lagrange interpolation”. In: *IEEE Access* 9 (2021), pp. 57464–57474.

[146] C. Yang, X. Chen, and Y. Xiang. “Blockchain-based publicly verifiable data deletion scheme for cloud storage”. In: *Journal of Network and Computer Applications* 103 (2018), pp. 185–193.

[147] Y. Yu, L. Xue, Y. Li, X. Du, M. Guizani, and B. Yang. “Assured data deletion with fine-grained access control for fog-based industrial applications”. In: *IEEE Transactions on Industrial Informatics* 14.10 (2018), pp. 4538–4547.

[148] S. Kaushik and C. Gandhi. “Capability based outsourced data access control with assured file deletion and efficient revocation with trust factor in cloud computing”. In: *International Journal of Cloud Applications and Computing (IJCAC)* 10.1 (2020), pp. 64–84.

[149] J. Ma, M. Wang, J. Xiong, and Y. Hu. “CP-ABE-based secure and verifiable data deletion in cloud”. In: *Security and Communication Networks* 2021 (2021), pp. 1–14.

[150] E. Bacis, S. De Capitani di Vimercati, S. Foresti, S. Paraboschi, M. Rosa, and P. Samarati. “Mix&Slice: Efficient access revocation in the cloud”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. Vienna: ACM, 2016, pp. 217–228.

[151] F. Beato, M. Kohlweiss, and K. Wouters. “Scramble! your social network data”. In: *International Symposium on Privacy Enhancing Technologies Symposium*. Springer. USA: Springer, 2011, pp. 211–225.

[152] A.-M. Olteanu, K. Huguenin, I. Dacosta, and J.-P. Hubaux. “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. 2018, pp. 1–16.

[153] Y.-T. Huang, D.-L. Chiang, T.-S. Chen, S.-D. Wang, F.-P. Lai, and Y.-D. Lin. “Lagrange interpolation-driven access control mechanism: Towards secure and privacy-preserving fusion of personal health records”. In: *Knowledge-based systems* 236 (2022), p. 107679.

[154] Y. He, H. Wang, Y. Li, K. Huang, V. C. Leung, F. R. Yu, and Z. Ming. “An efficient ciphertext-policy attribute-based encryption scheme supporting collaborative decryption with blockchain”. In: *IEEE Internet of Things Journal* 9.4 (2021), pp. 2722–2733.

[155] V. Shoup and N. P. Smart. “Lightweight asynchronous verifiable secret sharing with optimal resilience”. In: *Journal of Cryptology* 37.3 (2024), p. 27.

[156] Darwish, Marwan Adnan and Markatou, Evangelia Anna and Smaragdakis, Georgios. “Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment”. In: *Proceedings of the 18th European Workshop on Systems Security (EuroSec)*. New York, NY, USA: Association for Computing Machinery, 2025, pp. 77–83.

[157] K. M. Ramokapane and A. Rashid. “ExD: Explainable Deletion”. In: *Proceedings of the 2023 New Security Paradigms Workshop*. 2023, pp. 34–47.

[158] D. Zheng, L. Xue, C. Yu, Y. Li, and Y. Yu. “Toward assured data deletion in cloud storage”. In: *IEEE Network* 34.3 (2020), pp. 101–107.

[159] M. Johnson, S. Egelman, and S. M. Bellovin. “Facebook and privacy: it’s complicated”. In: *Proceedings of the eighth symposium on usable privacy and security*. 2012, pp. 1–15.

[160] T. Schnitzler, C. Utz, F. M. Farke, C. Pöpper, and M. Dürmuth. “Exploring user perceptions of deletion in mobile instant messaging applications”. In: *Journal of Cybersecurity* 6.1 (2020), tyz016.

[161] M. Minaei, M. Mondal, and A. Kate. “Empirical understanding of deletion privacy: Experiences, expectations, and measures”. In: *31st USENIX Security Symposium (USENIX Security 22)*. 2022, pp. 3415–3432.

[162] J. Xiong, X. Liu, Z. Yao, J. Ma, Q. Li, K. Geng, and P. S. Chen. “A secure data self-destructing scheme in cloud computing”. In: *IEEE Transactions on Cloud Computing* 2.4 (2014), pp. 448–458.

[163] Y. Miao, X. Liu, K.-K. R. Choo, R. H. Deng, J. Li, H. Li, and J. Ma. “Privacy-preserving attribute-based keyword search in shared multi-owner setting”. In: *IEEE Transactions on Dependable and Secure Computing* 18.3 (2019), pp. 1080–1094.

[164] C. Yang, Y. Liu, Y. Ding, and Y. Wu. “Block-based fine-grained and publicly verifiable data deletion for cloud storage”. In: *Soft Computing* 28.21 (2024), pp. 12491–12506.

[165] European Parliament and Council. “Regulation (EU) 2016/679 of the European Parliament and of the Council”. In: *Regulation (eu)* 679 (2016), p. 2016.

[166] M. Krzysztofek. *GDPR: General Data Protection Regulation (EU) 2016/679: Post-reform Personal Data Protection in the European Union*. Kluwer Law International BV, 2018.

[167] J. Reardon, D. Basin, and S. Capkun. “Sok: Secure data deletion”. In: *2013 IEEE Symposium on Security and Privacy*. IEEE. 2013, pp. 301–315.

[168] K. S. Niksirat, D. Korka, Q. Jacquemin, C. Vanini, M. Humbert, M. Cherubini, S. Métille, and K. Huguenin. “Security and Privacy with Second-Hand Storage Devices: A User-Centric Perspective from Switzerland”. In: *Proceedings on Privacy Enhancing Technologies (PoPETs)* 2024.2 (2024), p. 22.

[169] S. L. Garfinkel and A. Shelat. “Remembrance of data passed: A study of disk sanitization practices”. In: *IEEE Security & Privacy* 1.1 (2003), pp. 17–27.

[170] J. Lee, S. Yi, J. Heo, H. Park, S. Y. Shin, and Y. Cho. “An Efficient Secure Deletion Scheme for Flash File Systems.” In: *J. Inf. Sci. Eng.* 26.1 (2010), pp. 27–38.

[171] C. Cachin, K. Haralambiev, H.-C. Hsiao, and A. Sorniotti. “Policy-based secure deletion”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 259–270.

[172] Y. Luo, M. Xu, S. Fu, and D. Wang. “Enabling assured deletion in the cloud storage by overwriting”. In: *Proceedings of the 4th ACM international workshop on security in cloud computing*. 2016, pp. 17–23.

[173] B. Singh, R. Saharan, G. Somani, and G. Gupta. “Secure file deletion for solid state drives”. In: *Advances in Digital Forensics XII: 12th IFIP WG 11.9 International Conference, New Delhi, January 4-6, 2016, Revised Selected Papers 12*. Springer. 2016, pp. 345–362.

[174] D. Perito and G. Tsudik. “Secure code update for embedded devices via proofs of secure erasure”. In: *Computer Security–ESORICS 2010: 15th European Symposium on Research in Computer Security, Athens, Greece, September 20-22, 2010. Proceedings 15*. Springer. 2010, pp. 643–662.

[175] J. Tian and T. Zhang. “Secure and effective assured deletion scheme with orderly overwriting for cloud data”. In: *The Journal of Supercomputing* 78.7 (2022), pp. 9326–9354.

[176] M. Wei, L. Grupp, F. E. Spada, and S. Swanson. “Reliably Erasing Data From Flash-Based Solid State Drives”. In: *9th USENIX Conference on File and Storage Technologies (FAST 11)*. 2011.

[177] C. Liu, H. A. Khouzani, and C. Yang. “Erasucrypto: A light-weight secure data deletion scheme for solid state drives”. In: *Proceedings on Privacy Enhancing Technologies* (2017).

[178] V. Veeramachaneni. “Integrating Zero Trust Principles into IAM for Enhanced Cloud Security”. In: *Recent Trends in Cloud Computing and Web Engineering* 7.1 (2025), pp. 78–92.

[179] E. Stefanov and E. Shi. “Multi-cloud oblivious storage”. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013, pp. 247–258.

[180] X. Li, X. Wu, and J. Ni. “Accelerating Secure and Verifiable Data Deletion in Cloud Storage via SGX and Blockchain”. In: *GLOBECOM 2024-2024 IEEE Global Communications Conference*. IEEE. 2024, pp. 4138–4143.

[181] R. Kang, L. Dabbish, N. Fruchter, and S. Kiesler. “Data Just Goes Everywhere: User Mental Models of the Internet and Implications for Privacy and Security”. In: *Eleventh symposium on usable privacy and security (SOUPS 2015)*. 2015, pp. 39–52.

[182] A. Wittig and M. Wittig. *Amazon Web Services in Action: An in-depth guide to AWS*. Simon and Schuster, 2023.

[183] M. Collier and R. Shahan. *Microsoft azure essentials-fundamentals of azure*. Microsoft Press, 2015.

[184] E. Bisong and E. Bisong. “An overview of google cloud platform services”. In: *Building Machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners* (2019), pp. 7–10.

[185] R. Hussein, L. Scherdel, F. Nicolet, and F. Martin-Sanchez. “Towards the European Health Data Space (EHDS) ecosystem: A survey research on future health data scenarios”. In: *International Journal of Medical Informatics* 170 (2023), p. 104949.

[186] A. Demopoulos. *There are no serious safeguards’: can 23andMe be trusted with our DNA?* Accessed: 2025-04-03. url: https://www.theguardian.com/technology/2024/feb/17/23andme-dna-data-security-finance.

[187] L. Jamali. *23andMe users struggle to delete their highly sensitive data*. Accessed: 2025-04-01. url: https://www.bbc.com/news/articles/cddy8d63262o.

[188] S. Mavrovouniotis and M. Ganley. “Hardware security modules”. In: *Secure Smart Embedded Devices, Platforms and Applications*. Springer, 2013, pp. 383–405.

[189] I. Anati, S. Gueron, S. Johnson, and V. Scarlata. “Innovative technology for CPU based attestation and sealing”. In: *Proceedings of the 2nd international workshop on hardware and architectural support for security and privacy*. Vol. 13. 7. ACM New York, NY, USA. 2013.

[190] Chandramouli, Ramaswamy and Hibbard, Eric. *Guidelines for Media Sanitization*. Tech. rep. National Institute of Standards and Technology, 2025.

[191] C. Liu, R. Ranjan, C. Yang, X. Zhang, L. Wang, and J. Chen. "MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud". In: *IEEE Transactions on Computers* 64.9 (2014), pp. 2609–2622.

[192] S. Turner. "Transport Layer Security". In: *IEEE Internet Computing* 18.6 (2014), pp. 60–63.

[193] S. Boboň. "Analysis of NIST FIPS 140-2 Security Certificates". In: *Masaryk University: Brno, Czech Republic* (2021).

[194] S. Micali, M. Rabin, and J. Kilian. "Zero-knowledge sets". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE. 2003, pp. 80–91.

[195] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2022.

[196] M. Van Dijk, A. Juels, A. Oprea, R. L. Rivest, E. Stefanov, and N. Triandopoulos. "Hourglass schemes: how to prove that cloud files are encrypted". In: *Proceedings of the 2012 ACM conference on Computer and Communications security*. 2012, pp. 265–280.

[197] E. A. Luengo, B. A. Olivares, L. J. G. Villalba, and J. Hernandez-Castro. "Further analysis of the statistical independence of the NIST SP 800-22 randomness tests". In: *Applied Mathematics and Computation* 459 (2023), p. 128222.

[198] S. Contini, A. K. Lenstra, and R. Steinfeld. "VSH, an efficient and provable collision-resistant hash function". In: *Advances in Cryptology-EUROCRYPT 2006: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28-June 1, 2006. Proceedings 25*. Springer. 2006, pp. 165–182.

[199] A. Jha and M. Nandi. "A survey on applications of H-technique: Revisiting security analysis of PRP and PRF". In: *Entropy* 24.4 (2022), p. 462.

[200] M. Bellare and P. Rogaway. "Random oracles are practical: A paradigm for designing efficient protocols". In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*. 1993, pp. 62–73.

[201] AWS. *AWS CloudHSM*. Accessed: 2025-02-28. url: https://aws.amazon.com/cloudhsm/.

[202] AWS. *AWS Nitro Enclaves*. Accessed: 2025-02-28. url: https://aws.amazon.com/ec2/nitro/nitro-enclaves/.

[203] AWS. *AWS Key Management Service (KMS)*. Accessed: 2025-04-11. url: https://aws.amazon.com/kms/.

[204] AWS. *AWS Lambda*. Accessed: 2025-04-11. url: https://aws.amazon.com/lambda/.

[205] AWS. *Amazon S3 Object Storage*. Accessed: 2025-04-11. url: https://aws.amazon.com/s3/.

[206] AWS. *Amazon DynamoDB*. Accessed: 2025-04-11. url: https://aws.amazon.com/dynamodb/.

[207] AWS. *Amazon EC2 Instance Types*. Accessed: 2025-04-11. url: https://aws.amazon.com/ec2/instance-types/.

[208] AWS. *Amazon CloudWatch*. Accessed: 2025-04-11. url: https://aws.amazon.com/cloudwatch/.

[209] H. Gilbert and H. Handschuh. "Security analysis of SHA-256 and sisters". In: *International workshop on selected areas in cryptography*. Springer. 2003, pp. 175–193.

[210] M. Vaidehi and B. J. Rabi. "Design and analysis of AES-CBC mode for high security applications". In: *Second International Conference on Current Trends In Engineering and Technology-ICCTET 2014*. IEEE. 2014, pp. 499–502.

[211] E. Kiltz, A. O'Neill, and A. Smith. "Instantiability of RSA-OAEP under chosen-plaintext attack". In: *Journal of Cryptology* 30.3 (2017), pp. 889–919.

[212] Health Catalyst. *Healthcare Regulatory Measures Support Optimal Care*. https://shorturl.at/n71qf. Accessed: 2025-07-27.

[213] N. Scope, A. Rasin, B. Lenard, and J. Wagner. "Compliance and data lifecycle management in databases and backups". In: *International Conference on Database and Expert Systems Applications*. Springer. 2023, pp. 281–297.

[214] H. Gao, C. Yue, T. T. A. Dinh, Z. Huang, and B. C. Ooi. "Enabling secure and efficient data analytics pipeline evolution with trusted execution environment". In: *Proceedings of the VLDB Endowment* 16.10 (2023), pp. 2485–2498.

[215] A. Gutmann and M. Warner. "Fight to be forgotten: Exploring the efficacy of data erasure in popular operating systems". In: *Privacy Technologies and Policy: 7th Annual Privacy Forum, APF 2019, Rome, Italy, June 13–14, 2019, Proceedings 7*. Springer. 2019, pp. 45–58.

[216] H. Habib, Y. Zou, A. Jannu, N. Sridhar, C. Swoopes, A. Acquisti, L. F. Cranor, N. Sadeh, and F. Schaub. “An empirical analysis of data deletion and Opt-Out choices on 150 websites”. In: *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*. 2019, pp. 387–406.

[217] Statista. *How to delete yourself from the internet with 5 different methods*. https://www.statista.com/statistics/1172927/gdpr-obligation-difficultyamong-eu-and-us-firms/. Accessed: 2023-04-25.

[218] L. Bourtoule, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot. “Machine unlearning”. In: *2021 IEEE symposium on security and privacy (SP)*. IEEE. 2021, pp. 141–159.

[219] H. Hu, S. Wang, T. Dong, and M. Xue. “Learn what you want to unlearn: Unlearning inversion attacks against machine unlearning”. In: *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2024, pp. 3257–3275.

[220] H. Hu, S. Wang, J. Chang, H. Zhong, R. Sun, S. Hao, H. Zhu, and M. Xue. “A Duty to Forget, a Right to Be Assured? Exposing Vulnerabilities in Machine Unlearning Services”. In: *31st Annual Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2024.

[221] W. Wang, Z. Tian, C. Zhang, and S. Yu. “Machine unlearning: A comprehensive survey”. In: *arXiv preprint arXiv:2405.07406* (2024).

ACKNOWLEDGEMENTS

The sky is the limit.

From the ancient city of Aleppo — a cradle of civilization — to the PhD podium at Delft University of Technology, this journey has been one of endurance, hope, and an unwavering dedication to knowledge.

In 2017, as war engulfed my homeland and uncertainty shadowed every step, I left Syria carrying little more than determination and a fragile dream — that one day, I might stand again in peace, continuing the education I had begun amid chaos. What was once a distant hope became a purpose: to rebuild through learning, to rise through perseverance, and to transform hardship into progress.

This journey, however, has never been mine alone. Along the way, I was uplifted by the guidance of mentors, the steadfast love of family, and the quiet strength of those who believed in me — each shaping this research and the person behind it.

I would like to express my sincere gratitude to **Prof. Georgios Smaragdakis**, **Dr. Evangelia Anna Markatou**, and **Prof. Mauro Conti**, whose guidance shaped this journey.

George, thank you for your constant support, motivation, and belief in my potential throughout these four years. I especially appreciate your steady presence during the Go/No-Go phase and your willingness to take over my supervision for nearly a year — in addition to your role as promotor — during a critical transition. Your frequent feedback, thought-provoking discussions, and steady guidance shaped my week-to-week progress and the significant milestones of this research. I have grown profoundly under your mentorship, and your influence will remain with me beyond this PhD.

Lilika, thank you for stepping in as a co-promotor of my work during the second half of my PhD. Your thoughtful questions, sharp feedback, and commitment helped me complete this thesis and produce three substantial research contributions. You continually pushed me to improve — especially in my writing and critical thinking — and your mentorship has left a lasting mark on how I approach research.

Mauro, thank you for your thoughtful input during the Go/No-Go and annual assessment meetings, as well as for your insightful comments while I was completing the thesis. Your feedback has been instrumental in shaping and strengthening the final result.

I am equally thankful to the members of my doctoral committee —

Prof. Inald Lagendijk, Prof. Hamed Haddadi, Prof. Nikolaos Laoutaris, Dr. Seda Gürses, and Dr. Gunes Acar. Thank you for dedicating your time and expertise to reviewing my work and for attesting to the rigor and contribution of this dissertation. Your constructive feedback and thoughtful questions played a crucial role in shaping the final outcome of this research.

My deepest thanks go to my parents, whose unwavering love and lifelong sacrifices have laid the foundation for everything I have achieved.

To my father, **Adnan Darwish Khabbaz**, whose passion for knowledge and strength of character have always inspired me — you were my earliest role model in pursuing learning and living with purpose. You did not just encourage my education — you embodied it, lived it, and passed it on. I will never forget your words on the day I left Aleppo: “Do not return without achieving something great — come back victorious.” Those words became a compass for me and never left my heart.

To my mother, **Ghsoun Fattal**, whose quiet strength, endless prayers, and unconditional love carried me through every trial. You gave without asking, sacrificed without hesitation, and believed in me in ways I will never be able to repay. Your love was my refuge, your faith my anchor, and your presence — even from afar — the voice that reminded me to keep going.

To my wife, my love, my partner in all things — **Ola Hamza** — Your unwavering support, quiet strength, and limitless patience have upheld me throughout this PhD journey. You stood beside me through long nights, missed moments, and silent stresses — often giving more than you had, so I could give more to my work. Your faith in me never wavered, even when mine did. Thank you for reminding me that beyond every paper and deadline, there is life, love, and something greater waiting at home. I am endlessly grateful that you are my partner in this chapter, and every one to come.

To my precious children, **Diala and Adnan** — You are still too young to understand what this journey represents, but you are already the greatest gift life has given me. Your laughter lifted my spirit on the hardest days, and your presence gave this work meaning far beyond academia. I pray that the world opens wide for you — and that you will one day chase your dreams with the same hope and fire that your little hearts helped me carry. Fly high, my dreaming birds — the sky is yours.

I would also like to extend my heartfelt thanks to the rest of my family and my dear friends, whose presence and encouragement have meant more than words can express.

I am grateful to my colleagues in the Cybersecurity Research Group at TU Delft for the stimulating discussions, collaboration, and supportive environment. I also extend my thanks to my colleagues at the University of Kuala Lumpur and to Malaysia. This country welcomed me and

allowed me to take my first steps into academia.

I remain especially thankful to Aleppo University, where the spark of this journey was first ignited — where curiosity turned into purpose, and my passion for knowledge found its earliest direction.

Though years and borders now separate me from my homeland, Syria has never left my heart. Since I last saw Aleppo in 2017, not a single day has passed without its streets, people, and spirit living within me. Every page of this thesis, every hour of work, and every challenge overcome has been, in some quiet way, a tribute to the country that shaped me — a message sent from afar: *I have not forgotten*.

This PhD is more than an academic achievement — it is a humble offering of gratitude to Syria and Aleppo, where my earliest dreams were born. May it stand as a small testament to the strength of those who endure and the hope that still rises from the ruins.

CURRICULUM VITæ

M-Marwan Darwish Khabbaz

10-01-1993	Born in Aleppo, Syria.

EDUCATION

Present	Postdoc Fellowship Radboud University, The Netherlands
2021–2025	Ph.D. Candidate Delft University of Technology, The Netherlands
2017–2019	M.Sc. in Information Technology Kuala Lumpur University (UNIKL), Malaysia
2016–2017	M.Sc. in Web Sciences Syrian Virtual University (SVU), Syria
2010–2016	B.Sc. in Informatics Engineering Aleppo University, Syria

LIST OF PUBLICATIONS

CONFERENCE AND WORKSHOP

3. **M. A. Darwish**, E. A. Markatou, and G. Smaragdakis. *Provable Co-Owned Data Deletion with Zero-Residuals and Verifiability in Multi-Cloud Environment*. In Proceedings of the 18th European Workshop on Systems Security (EuroSec), 2025, pp. 77–83. https://dl.acm.org/doi/abs/10.1145/3722041.3723104

2. **M. A. Darwish** and G. Smaragdakis. *Disjunctive Multi-Level Digital Forgetting Scheme*. In Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing (SAC), 2024, pp. 112–121. https://dl.acm.org/doi/abs/10.1145/3605098.3635904

1. **M. A. Darwish** and A. Zarras. *Digital Forgetting Using Key Decay*. In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC), 2023, pp. 34–41. https://dl.acm.org/doi/abs/10.1145/3555776.3577641

MANUSCRIPTS CURRENTLY UNDER REVIEW

2. **M. A. Darwish**, E. A. Markatou, and G. Smaragdakis. *A Policy-Based Conjunctive Scheme for Digital Forgetting of Co-Owned Data*. Under review, 2025.

1. **M. A. Darwish**, E. A. Markatou, and G. Smaragdakis. *From Creation to Deletion: Secure and Verifiable Data Management in Cloud Environments*. Under review, 2025.

OTHER PUBLICATION DURING THE PHD

1. K. Lampropoulos, A. Zarras, E. Lakka, P. Barmpaki, K. Drakonakis, M. Athanatos, H. Debar, A. Alexopoulos, A. Sotiropoulos, G. Tsakirakis, and **M. A. Darwish**. *White Paper on Cybersecurity in the Healthcare Sector: The HEIR Solution*. arXiv preprint: arXiv:2310.10139, 2023. https://arxiv.org/abs/2310.10139


TUDelft