Searchable Encryption: Building Practical and Secure Cloud Data Systems

by

Jiaming Yuan

A dissertation accepted and approved in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computer & Information Science

Dissertation Committee:
Yingjiu Li, Chair
Jun Li, Core Member
Ram Durairajan, Core Member
Jon Brundan, Institutional Representative

University of Oregon
Winter 2026

DISSERTATION ABSTRACT

Jiaming Yuan

Doctor of Philosophy in Computer & Information Science

Title: Searchable Encryption: Building Practical and Secure Cloud Data Systems

The rapid adoption of cloud computing has revolutionized data storage and accessibility but has simultaneously introduced profound privacy challenges. While encryption ensures data confidentiality, it inherently renders traditional search functionalities obsolete, creating a tension between security and utility. Searchable Encryption (SE) emerged to bridge this gap; however, existing SE solutions often struggle to balance high efficiency, robust security against leakage-abuse attacks, and decentralized trust management in practical, multi-domain environments.

This dissertation addresses these critical limitations by proposing a comprehensive framework for practical and secure searchable encryption. We first provide a systematic taxonomy and survey of registration-based cryptography, tracing the evolution from fully-trusted authorities to semi-trusted curators, which establishes the theoretical foundation for removing the key escrow problem in search systems.

Second, we address security vulnerabilities in symmetric settings by introducing the Volume Leakage-Resilient Easily Deployable and Efficiently Searchable Encryption (VLR-EDESE) scheme. By leveraging k-indistinguishability and obfuscation, VLR-EDESE effectively mitigates query and document volume leakage—a primary target for leakage-abuse attacks—while maintaining operational simplicity for standard cloud storage providers.

Third, to support complex query logic without relying on a centralized trusted authority, we propose RBAE2KS, a registration-based authenticated encryption scheme.

RBAE2KS integrates vector commitments and Linear Secret Sharing Schemes (LSSS) to enable expressive searches over monotone Boolean formulas while remaining resilient against offline keyword guessing attacks.

Finally, we bridge the gap between cryptographic design and real-world deployment with M-EDESE, the first multi-domain searchable encryption system designed for inter-domain partnerships. M-EDESE allows cross-domain keyword queries with high efficiency and privacy, requiring zero additional cooperation from the cloud storage provider. Extensive experimental evaluations demonstrate that the proposed protocols achieve a superior balance of security and performance, providing a scalable pathway toward the deployment of secure cloud data systems in sensitive sectors such as healthcare and finance.

This dissertation includes published and unpublished co-authored materials.

CURRICULUM VITAE

NAME OF AUTHOR: Jiaming Yuan

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
China University of Petroleum, China

DEGREES AWARDED:

Doctor of Philosophy, Computer & Information Science, 2026 (Expected), University of Oregon
Master of Science, Computer & Information Science, 2024, University of Oregon
Bachelor of Science, Software Engineering, 2015, China University of Petroleum

AREAS OF SPECIAL INTEREST:

Searchable Encryption
Attribute-based Encryption
Registration-based Encryption

PROFESSIONAL EXPERIENCE:

Research Assistant, Center for Cyber Security and Privacy, University of Oregon, 2025-Present
Teaching Assistant (CS 122 Intro to Programming, CS 333 Applied Cryptography, CS 436/536 Secure Software Development, CS 432/532 Introduction to Networks), University of Oregon, 2023-2024
Research Assistant, Center for Cyber Security and Privacy, University of Oregon, 2021-2023
Research Engineer, Secure Mobile Center, Singapore Management University, 2016-2021
Software Engineer, Realtime Invention (Beijing), 2016

GRANTS, AWARDS AND HONORS:

Lorry I. Lokey Interdisciplinary Science Initiative Dissertation Award, University of Oregon, 2026

Gurdeep Pall Graduate Student Fellowship, University of Oregon, 2025

Runner-up Poster Award, Graduate Research Open House, University of Oregon, 2025

Ripple Fellowship, University of Oregon, 2022

PUBLICATIONS:

Yuan, J., Li, Y., Deng, R. H., Ning, J., & Xu, S. (2025). RBAE^{2}KS: Registration-Based Authenticated Encryption with Expressive Keyword Search. *In submission*.

Yuan, J., Li, Y., Yang, Y., Wu, D., Ning, J., Xu, S., & Deng, R. H. (2025). From Fully-Trusted Private Key Generator to Semi-Trusted Key Curator: A Survey on Registered Encryption. *In submission*.

Yuan, J., Li, Y., Li, J., Wu, D., Ning, J., Tian, Y., & Deng, R. H. (2025). Leakage-Resilient Easily Deployable and Efficiently Searchable Encryption (EDESE). *ACM Symposium on Access Control Models and Technologies (SACMAT 2025)*, 133–144.

Li, Y., Sengupta, B., Tian, Y., Yuan, J., & Yuen, T. H. (2014). Message Control for Blockchain Rewriting. *IEEE Transactions on Dependable and Secure Computing*, 22(3), 1864–1876.

Xu, S., Ning, J., Li, X., Yuan, J., Huang, X., & Deng, R. H. (2024). A Privacy-Preserving and Redactable Healthcare Blockchain System. *IEEE Transactions on Services Computing*, 17(2), 364–377.

Xu, S., Huang, X., Yuan, J., Li, Y., & Deng, R. H. (2022). Accountable and Fine-Grained Controllable Rewriting in Blockchains. *IEEE Transactions on Information Forensics and Security*, 18, 101–116.

Yuan, J., Li, Y., Ning, J., & Deng, R. H. (2022). M-EDESE: Multi-Domain, Easily Deployable, and Efficiently Searchable Encryption. *International Conference on Information Security Practice and Experience (ISPEC 2022)*, 606–623.

Zhao, B., Yuan, J., Liu, X., Wu, Y., Pang, H., & Deng, R. H. (2022). SOCI: A Toolkit for Secure Outsourced Computation on Integers. *IEEE Transactions on Information Forensics and Security*, 17, 3637–3648.

Ning, J., Huang, X., Poh, G. S., Yuan, J., Li, Y., Weng, J., & Deng, R. H. (2021). LEAP: Leakage-Abuse Attack on Efficiently Deployable, Efficiently Searchable Encryption with Partially Known Dataset. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS 2021)*, 2307–2320.

Hou, H., Hao, S., Yuan, J., Xu, S., & Zhao, Y. (2021). Fine-Grained and Controllably Redactable Blockchain with Harmful Data Forced Removal. *Security and Communication Networks*, 2021, 3680359.

Xu, S., Ning, J., Ma, J., Xu, G., Yuan, J., & Deng, R. H. (2021). Revocable Policy-Based Chameleon Hash. *Computer Security – ESORICS 2021*, 327–347.

Yuan, H., Chen, X., Wang, J., Yuan, J., Yan, H., & Susilo, W. (2020). Blockchain-Based Public Auditing and Secure Deduplication with Fair Arbitration. *Information Sciences*, 541, 409–425.

Ning, J., Cao, Z., Dong, X., Yuan, J., Liang, K., Ma, H., & Wei, L. (2020). Efficient Ciphertext-Policy Attribute-Based Encryption with Blackbox Traceability. *Information Sciences*, 528, 218–232.

Sun, J., Xiong, H., Zhang, S., Liu, X., Yuan, J., & Deng, R. H. (2020). A Secure Flexible and Tampering-Resistant Data Sharing System for Vehicular Social Networks. *IEEE Transactions on Vehicular Technology*, 69(11), 12938–12950.

Xu, S., Yuan, J., Li, Y., Liu, X., & Zhang, Y. (2019). Super Payment Channel for Decentralized Cryptocurrencies. *IEEE Conference on Dependable and Secure Computing (DSC 2019)*, 1–8.

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Joe Li, for his transformative mentorship. Joe did far more than supervise my progress; he taught me how to truly "dig into" a topic, peeling back layers of complexity to find the fundamental questions worth asking. I am especially grateful for his guidance in the art of academic storytelling. His influence is visible on every page of my published work and this dissertation.

I am also sincerely grateful to my dissertation committee members: Dr. Jun Li, Dr. Ram Durairajan, and Dr. Jon Brundan. Their rigorous feedback and diverse insights were invaluable in helping me refine my ideas and push the boundaries of my research.

My thanks also go to my collaborators and co-authors. The papers we produced together were the highlights of my doctoral years, and I am grateful for the shared intellectual curiosity that drove those projects forward.

Finally, I owe everything to my family. Though we have been separated by thousands of miles, your love and support have known no distance. Your unwavering belief in me provided the emotional foundation I needed to navigate the challenges of a PhD. Every phone call and every word of encouragement from home provided the strength I needed to persevere. This achievement is as much yours as it is mine. Thank you for your endless patience, love, and for always being my greatest source of motivation.

TABLE OF CONTENTS

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

The rapid proliferation of cloud computing has transformed how individuals and organizations manage their data, enabling ubiquitous access and cost-effective storage. However, outsourcing sensitive data to third-party cloud service providers (CSPs) introduces a fundamental tension between data utility and privacy. While data owners are incentivized to use the cloud for its vast computational and storage resources, storing information such as electronic health records (EHRs) or financial statements in plaintext exposes it to unauthorized access, insider threats, and legal subpoenas.

To mitigate these risks, regulatory frameworks like General Data Protection Regulation (GDPR) and Health Insurance Portability and Accountability Act (HIPAA) often mandate end-to-end encryption. While standard encryption (e.g., AES, RSA) ensures confidentiality, it creates a "data silo" effect: because the cloud provider cannot "see" the data, it cannot perform even basic operations like searching or filtering.

A naive solution to this problem is the "download-and-decrypt" approach, where a user downloads the entire encrypted database to a local machine, decrypts it, and performs a search. This approach, however, is often impractical. It imposes a massive bandwidth burden and requires the client to possess significant local storage and processing power, effectively nullifying the primary benefits of cloud outsourcing. Searchable Encryption (SE) emerged to solve this specific bottleneck by enabling keyword matching on encrypted data without revealing the underlying plaintext to the server.

The operational model of SE typically involves three phases:

1. **Setup and Encryption:** The data owner encrypts the documents and a searchable index locally before uploading them to the cloud.

2. **Trapdoor Generation:** To search for keyword(s), the authorized user generates a "trapdoor" (an encrypted search token) using a secret key.

3. **Cloud Search:** The CSP uses the trapdoor to identify matching encrypted documents and returns them to the user without ever learning the actual keyword or the document content.

Despite these advantages, bridging the gap between theoretical SE and a practical, secure system remains an open challenge. Current solutions suffer from bottlenecks in three types of environment settings, single-user settings, multi-user settings, and multi-domain collaboration settings.

The first bottleneck concerns leakage vulnerabilities in single-user settings. Symmetric Searchable Encryption (SSE) schemes are designed for single-user settings, where a data owner encrypts and queries their own dataset using a shared secret key. To improve practical usability, Easily-Deployable and Efficiently-Searchable Encryption (EDESE) schemes have been proposed, with no operations of the cloud server except binary matching. However, SSE schemes, including EDESE, inherently expose access patterns and volume patterns to the cloud server, making them susceptible to Leakage Abuse Attacks (LAAs). In such attacks, an adversary — typically a semi-honest CSP — exploits observable leakage, such as the number of matching documents returned per query, to reconstruct the underlying plaintext keywords with high accuracy. Empirical studies have demonstrated that even partial auxiliary knowledge of the dataset distribution suffices for a successful LAA, severely undermining the privacy guarantees of EDESE and limiting its applicability for protecting sensitive individual data in real-world cloud deployments.

The second bottleneck arises in multi-user settings. Public Key Encryption with Keyword Search (PEKS) extends searchable encryption to multi-user settings, enabling

any sender to encrypt data under a recipient's public key while allowing the recipient to delegate search capabilities to a server via trapdoors. However, PEKS is inherently vulnerable to Keyword Guessing Attacks (KGAs): since keyword spaces are typically small and predictable, an adversary in possession of a trapdoor can enumerate candidate keywords, generate test ciphertexts under the public key, and verify matches against the trapdoor without bound. This offline attack requires no interaction with the data owner and is computationally feasible even for modestly sized keyword spaces. Beyond KGAs, practical deployments demand that PEKS support expressive search predicates — such as conjunctive, disjunctive, and general Boolean queries — to accommodate complex real-world query workloads. Yet existing expressive PEKS constructions with KGA defense commonly rely on a fully-trusted Private Key Generator (PKG), which holds a master secret key and issues private keys to all users. This centralized trust model introduces a key-escrow problem: a compromised or malicious PKG can decrypt any ciphertext or forge any trapdoor in the system, creating a single point of failure that fundamentally conflicts with the privacy objectives of outsourced data management.

The third bottleneck stems from the limitations of existing schemes in multi-domain settings. Most SE schemes — both symmetric and asymmetric — are architected for single-domain environments, where all participants share a common trust anchor, key infrastructure, and system parameters. This assumption fails in federated or cross-organizational scenarios, such as inter-hospital health record sharing or cross-enterprise financial auditing, where data owners and data users belong to distinct administrative domains governed by independent authorities. Extending single-domain SE to multi-domain settings introduces non-trivial challenges: trust relationships between domains must be established without surrendering each domain's internal autonomy; cross-domain trapdoor verification must remain efficient without leaking intra-domain sensitive

information; and the system must tolerate the absence of a globally trusted third party. Current solutions lack a principled framework for reconciling these requirements, leaving multi-domain collaboration as a largely unaddressed problem in the searchable encryption literature.

1.1 Dissertation Statement

This dissertation posits that the trade-off between security, efficiency, and trust in searchable encryption can be reconciled through a paradigm shift toward decentralized trust architectures and leakage-resilient obfuscation. We argue that by integrating Registration-Based Encryption (RBE) and novel k-indistinguishable algorithms, it is possible to construct a suite of SE systems tailored for diverse environments that are: (1) resilient to leakage-abuse attacks in single-user settings; (2) free from key-escrow vulnerabilities in multi-user settings; and (3) capable of operating across independent administrative domains without a globally trusted third party.

1.2 Key Contributions

The primary contributions of this dissertation are organized into four major research efforts, each addressing a specific facet of the problems identified above.

1.2.1 Systematic Survey of Registration-Based Encryption (Chapter III).

We provide the first comprehensive survey that traces the evolution from fully-trusted PKGs to semi-trusted Key Curators, establishing the theoretical foundation for decentralized searchable encryption.

1.2.2 Leakage-Resilient Symmetric Searchable Encryption (Chapter IV).

We introduce a novel leakage-resilient scheme that protects against query and document volume leakage using k-indistinguishability, achieving superior security with significantly lower overhead than ORAM-based solutions.

1.2.3 Registration-Based Expressive Asymmetric Searchable Encryption

(Chapter V). We propose RBAE^{2}KS,a registration-based authenticated encryption framework that resolves the key-escrow problem and supports complex monotone Boolean formulas while remaining KGA-resistant.

1.2.4 Multi-Domain Deployable Searchable Systems (Chapter VI).

We design and implement the first multi-domain, easily-deployable SE system, facilitating cross-domain keyword queries without requiring additional coordination from cloud providers.

1.3 Dissertation Organization

The remainder of this dissertation is organized as follows: Chapter II provides a concise, problem-oriented review of the literature on searchable encryption and registration-based cryptography, serving as motivation for the four technical chapters that follow. Chapter III presents an independent academic contribution: a comprehensive survey on the taxonomy of RBE, providing technical support for decentralization in searchable encryption systems. Chapter IV describes the design and evaluation of our leakage-resilient symmetric scheme. Chapter V introduces the registration-based asymmetric framework for expressive searches. Chapter VI details the implementation of our multi-domain system. Finally, Chapter VII concludes the dissertation and discusses future research directions.

1.4 Co-authored Materials and Acknowledgments

In accordance with the Graduate School requirements of the University of Oregon, this section acknowledges the co-authored materials included in this dissertation.

1.4.1 Co-authored Materials.

Chapters IV, V, VI, and the survey in Chapter III contain material from the following co-authored manuscripts:

– **Chapter III** contains material from the manuscript titled *"From Fully-Trusted Private Key Generator to Semi-Trusted Key Curator: A Survey on Registered Encryption,"* which has been submitted for publication in 2026, authored by Jiaming Yuan, Yingjiu Li, Yang Yang, Daoyuan Wu, Jianting Ning, Shengmin Xu, and Robert H. Deng.

– **Chapter IV** is based on the paper *"Leakage-Resilient Easily Deployable and Efficiently Searchable Encryption (EDESE),"* published in the *Proceedings of the 30th ACM Symposium on Access Control Models and Technologies (SACMAT 2025)*, authored by Jiaming Yuan, Yingjiu Li, Jun Li, Daoyuan Wu, Jianting Ning, Yangguang Tian, and Robert H. Deng.

– **Chapter V** contains material from the manuscript titled *"RBAE2KS: Registration-Based Authenticated Encryption with Expressive Keyword Search,"* which is currently submitted for publication in 2026, authored by Jiaming Yuan, Yingjiu Li, Robert H. Deng, Jianting Ning, and Shengmin Xu.

– **Chapter VI** is based on the paper *"M-EDESE: Multi-Domain, Easily Deployable, and Efficiently Searchable Encryption,"* published in the *Proceedings of the 17th International Conference on Information Security Practice and Experience (ISPEC 2022)*, authored by Jiaming Yuan, Yingjiu Li, Jianting Ning, and Robert H. Deng.

CHAPTER II

LITERATURE REVIEW

The field of Searchable Encryption (SE) has evolved from basic symmetric constructions to complex, trust-minimized asymmetric frameworks. This chapter provides a comprehensive review of the existing literature, drawing from foundational theories and contemporary advancements in both symmetric and asymmetric settings.

2.1 Symmetric Searchable Encryption (SSE) and Leakage

Symmetric Searchable Encryption (SSE) allows users to outsource encrypted data while retaining search capabilities using symmetric keys, e.g., Curtmola, Garay, Kamara, and Ostrovsky (2006); Song, Wagner, and Perrig (2000).

2.1.1 Structured Encryption and Index Retrieval.

Many modern SSE schemes are categorized as Structured Encryption (STE) like Chase and Kamara (2010), where relationships between keywords and documents are represented as encrypted multi-maps (EMMs). While STE secures the index retrieval phase—allowing users to retrieve document indices matching a searched keyword—it introduces potential vulnerabilities during the subsequent document retrieval phase. As demonstrated in recent studies like George, Kamara, and Moataz (2021), if an SSE system protects only the search index but naively outsources encrypted documents, attackers can observe the retrieved documents and infer underlying index patterns.

2.1.2 Limitations in Defending Leakage-Abuse Attacks (LAA).

A significant portion of SSE research focuses on mitigating Leakage-Abuse Attacks (LAAs), which exploit search and access patterns. However, existing countermeasures often prove inadequate for practical cloud deployments:

- **ORAM-based Schemes:** Approaches utilizing Oblivious RAM introduced in S. Garg, Mohassel, and Papamanthou (2016); Wu and Li (2023) offer strong

security but introduce prohibitive overhead in computation, server-side state maintenance, and communication.

- **Deployment Bottlenecks:** Many schemes require complex server-side operations that go beyond the fundamental "storage and matching" capabilities of standard cloud providers.

This gap necessitates the design of easily deployable schemes that provide robust protection for both index and document retrieval, a challenge we address in Chapter IV.

2.2 Public Key Encryption with Keyword Search (PEKS)

Public Key Encryption with Keyword Search (PEKS) first introduced by Boneh, Di Crescenzo, Ostrovsky, and Persiano (2004a) extends search capabilities to multi-user environments, but it faces inherent security and functional limitations.

2.2.1 Expressive Asymmetric Searchable Encryption (EASE).

Foundational PEKS schemes were primarily limited to exact keyword matching, which curtails their utility in complex real-world scenarios. This spurred the development of Expressive Asymmetric Searchable Encryption (EASE), e.g., Shen, Lu, and Li (2019); Q. Xu et al. (2021), which supports conjunctive, disjunctive, and range searches. However, as functionality increases, so does the complexity of maintaining security against adaptive adversaries.

2.2.2 Keyword Guessing Attacks (KGA). A critical vulnerability in PEKS is the Offline Keyword Guessing Attack (KGA). Since keywords often follow predictable distributions, an adversary can use the public key to generate ciphertexts for candidate keywords and compare them against intercepted trapdoors. While Identity-Based Authenticated Encryption with Keyword Search (IBAEKS) was proposed to mitigate KGA, most existing solutions still rely on a fully-trusted authority.

2.3 The Registration-Based Encryption (RBE) Paradigm

To resolve the tension between identity-based management and the risks of centralized trust, the Registration-Based Encryption (RBE) paradigm was introduced by S. Garg, Hajiabadi, Mahmoody, and Rahimi (2018).

2.3.1 Decentralizing the Private Key Generation. Unlike traditional Identity-Based Encryption (IBE) where a Private Key Generator (PKG) holds the master secret, RBE introduces a semi-trusted **Key Curator (KC)** or Registration Server. In this model:

- Users generate their own public/private key pairs locally.
- The KC only validates and registers public keys within a compact commitment (e.g., a Merkle tree).

This decentralization ensures that no single entity can unilaterally reconstruct a user's private key, effectively eliminating the key escrow problem.

2.3.2 Challenges in Expressive Searchable RBE. While RBE addresses the escrow issue, integrating it into searchable encryption—especially those requiring expressive queries (monotone Boolean formulas)—introduces intricate challenges. It requires balancing the efficiency of RBE registration with the complexity of Linear Secret Sharing Schemes (LSSS) and vector commitments, which we explore in Chapter V.

2.4 Multi-Domain and Searchable Systems in Practice

While foundational SE research primarily focuses on single-owner or single-domain settings, real-world cloud applications—especially in healthcare and cross-organizational collaboration—demand a multi-domain architecture.

2.4.1 Multi-User and Cross-Domain Challenges. Traditional Multi-User Searchable Encryption (MUSE) schemes like Bao, Deng, Ding, and Yang (2008) allow

a data owner to share search capabilities with authorized users. However, extending this to a *multi-domain* setting, where multiple independent entities (e.g., different hospitals or branches) manage their own datasets and user bases, introduces significant hurdles. Most existing cross-domain solutions require either a centralized trusted authority to manage inter-domain keys or a heavy collaboration between the cloud storage providers (CSPs) of different domains.

2.4.2 System Deployability and Practical Constraints. A recurring gap in SE literature is the tension between theoretical security and "Easily-Deployable" (ED) characteristics. Many advanced schemes require the cloud server to perform complex cryptographic operations (e.g., pairing-based matching or secret sharing reconstruction) that are not supported by standard cloud storage APIs like Amazon S3 or Google Cloud Storage.

- **Operational Complexity:** Schemes requiring the server to maintain a persistent state or execute non-standard scripts are often rejected by commercial CSPs due to security and scalability concerns.
- **Inter-Domain Partnerships:** In practical scenarios, users from Domain A should be able to query records in Domain B based on pre-established partnerships without re-encrypting the entire database or relying on a global root of trust.

2.4.3 Transition to M-EDESE. The need for a system that supports cross-domain keyword queries with high efficiency, while remaining content-agnostic and requiring zero additional cooperation from the CSP, remains a critical research objective. This dissertation addresses this gap in Chapter VI by introducing the M-EDESE framework, which leverages efficient token transformation techniques to enable privacy-preserving multi-domain search without sacrificing deployability.

2.5 Summary

This chapter has provided a comprehensive synthesis of the evolution of Searchable Encryption, highlighting the fundamental tension between operational efficiency, cryptographic security, and trust decentralization. We first reviewed the landscape of **Symmetric Searchable Encryption (SSE)**, noting that while it offers high performance, it remains inherently vulnerable to leakage-abuse attacks during both index and document retrieval—a gap that necessitates the leakage-resilient mechanisms proposed in **Chapter IV**.

Moving to the asymmetric setting, we analyzed how **Public Key Encryption with Keyword Search (PEKS)** facilitates multi-user environments but introduces the critical vulnerabilities of key escrow and keyword guessing attacks. The emergence of the **Registration-Based Encryption (RBE)** paradigm offers a transformative solution by shifting trust from an all-powerful PKG to a semi-trusted Key Curator. However, as we have identified, integrating RBE with expressive query capabilities and ensuring its scalability in multi-domain, real-world cloud infrastructures remains an unresolved challenge.

By synthesizing these diverse research threads—ranging from volume-hiding techniques in SSE to decentralized trust in RBAE2KS and cross-domain interoperability in M-EDESE—this dissertation seeks to establish a holistic framework for secure cloud data systems. The subsequent chapters will detail our specific contributions in addressing these identified limitations, moving toward a future of practical, expressive, and trust-minimized searchable encryption.

CHAPTER III

FROM FULLY-TRUSTED PRIVATE KEY GENERATOR TO SEMI-TRUSTED KEY CURATOR: A SURVEY ON REGISTERED ENCRYPTION

This chapter contains material from the manuscript titled *"From Fully-Trusted Private Key Generator to Semi-Trusted Key Curator: A Survey on Registered Encryption,"* which has been submitted for publication. I was the primary author, responsible for the taxonomy,literature synthesis, and manuscript preparation. Dr. Yingjiu Li provided conceptual guidance and editorial revisions. Yang Yang, Daoyuan Wu, Jianting Ning, Shengmin Xu, and Robert H. Deng contributed to the refinement of the manuscript.

We conducted a comprehensive literature review on RBE, examining its foundational principles, various constructions, and the security properties it achieves. We also analyzed how RBE has been applied in different cryptographic contexts and identified the challenges and limitations associated with its integration into complex primitives like searchable encryption. This review provided critical insights that informed our design of an RBE-based IBAEKS scheme, ensuring that we effectively leverage RBE's strengths while addressing its challenges in the context of secure keyword search.

3.1 Introduction

The growing need for secure and decentralized data-sharing systems has catalyzed significant interest in advanced cryptographic primitives that enable fine-grained access control. Identity-Based Encryption (IBE), Attribute-Based Encryption (ABE), and Functional Encryption (FE) provide sophisticated mechanisms for enforcing expressive access policies on encrypted data, facilitating privacy-preserving applications in cloud computing, collaborative platforms, and Internet of Things (IoT) environments. However, despite their theoretical appeal and cryptographic elegance, widespread adoption of these

primitives has been constrained by fundamental trust assumptions that are incompatible with contemporary privacy and security imperatives.

These schemes face significant deployment barriers due to the key escrow problem. In traditional implementations, a central authority—the Private Key Generator (PKG) maintains the ability to generate decryption keys for any user based on their identities, attributes, or authorized functions. While this centralized approach simplifies key management by eliminating traditional certificate authorities, it creates a critical vulnerability where the PKG can decrypt all system communications.

Key Escrow Issue. The key escrow problem transcends theoretical concerns, fundamentally undermining the confidentiality guarantees these systems promise to deliver. When a single entity possesses universal decryption capabilities, it introduces critical architectural vulnerabilities that threaten overall system integrity. The PKG is vulnerable to a single point of failure—its compromise jeopardizes the security of the entire network by potentially exposing all encrypted communications. This risk escalates with system adoption, making the PKG an increasingly attractive target for sophisticated adversaries.

Furthermore, the centralized trust model inherent in such systems stands in direct opposition to contemporary security principles, which emphasize decentralization, distributed trust, and the minimization of single points of authority. Users are required to place complete trust in the PKG's operational integrity and its resilience against both external threats and internal misuse.

This centralized trust paradigm proves particularly problematic for applications demanding robust privacy protections, including secure messaging platforms, confidential reporting mechanisms, and decentralized infrastructures where trust minimization is paramount.

Registration-Based Encryption: Shift Trust to the Edge. To address these fundamental limitations, S. Garg et al. (2018) introduced Registration-Based Encryption (RBE)[1] as a paradigmatic shift in the identity-based encryption design. RBE fundamentally reconceptualizes the trust architecture by replacing the fully trusted PKG with a semi-trusted Key Curator (KC) that operates under significantly constrained authorities. This architectural transformation enables users to autonomously generate their public/secret key pairs and register their public keys deterministically under their respective identities, while the KC aggregates these public keys into evolving public parameters that facilitate encryption to specific identities without requiring knowledge of any secret keys.

The operational mechanics of RBE preserve the intuitive encryption model of traditional IBE, e.g., Boneh and Franklin (2001); Döttling and Garg (2017a); Shamir (1985), wherein encryption requires only the specification of the recipient's identity and access to current public parameters. However, the decryption process introduces a critical architectural distinction: successful decryption requires both the recipient's privately held secret key and a publicly computable helper key associated with their identity. This helper key must undergo periodic refreshment as public parameters evolve in response to new user registrations. Significantly, the KC's operational scope is limited to performing deterministic computations while maintaining no sensitive cryptographic state, rendering it auditable and reducing its threat profile to that of an honest-but-curious adversary.

This architectural innovation preserves the practical advantages of IBE—particularly its elimination of complex certificate management—while

[1] Since registration-based encryption (RBE) replaces the fully trusted PKG in IBE with a semi-trusted KC for registering users' public keys and identities, the term "registered IBE (RIBE)" might more precisely reflect its design, as "RBE" may overgeneralize its meaning. Nevertheless, we adopt the original terminology "RBE" as introduced in S. Garg et al. (2018) to maintain consistency with prior work.

fundamentally addressing its most critical security vulnerability: the concentration of universal decryption authority within a single trusted entity.

Beyond RBE: The Registered Encryption (RE) Paradigm. Building on the foundational ideas of RBE, researchers have generalized the registration-based approach into a broader class of encryption schemes collectively referred to as the *Registered Encryption (RE)* paradigm, e.g., Chiku, Hara, Hashimoto, Tomita, and Shikata (2024); Cong, Eldefrawy, and Smart (2021); Fiore, Kolonelos, and Perthuis (2023); S. Garg, Hajiabadi, Mahmoody, Rahimi, and Sekar (2019a); Glaeser et al. (2023a); R. Goyal and Vusirikala (2020); Hajiabadi, Mahmoody, Qi, and Sarfaraz (2023); Mahmoody, Qi, and Rahimi (2022); Q. Wang et al. (2023). These include Registered Attribute-Based Encryption (RABE) and Registered Functional Encryption (RFE), which adopt the semi-trusted KC model to extend the benefits of RBE—such as eliminating key escrow and enabling transparency—to richer access control settings.

RABE adapts the KC-based architecture to attribute-based encryption, particularly ciphertext-policy ABE (CP-ABE), e.g., Attrapadung and Tomida (2024); Freitag, Waters, and Wu (2023); R. Garg, Lu, Waters, and Wu (2024); Hohenberger, Lu, Waters, and Wu (2023); Y. Zhang, Zhao, Zhu, Gong, and Chen (2024); Zhu, Zhang, Gong, and Qian (2023). In this model, users generate their own public/secret key pairs and register their public keys along with attribute sets through the KC. When encrypting a message, the sender specifies an access policy over attributes. A user can decrypt the ciphertext only if their registered attributes satisfy this policy, using their secret key and a periodically refreshed helper key derived from evolving public parameters. RABE retains the expressiveness of ABE for fine-grained access control while removing the key escrow risk inherent in traditional architectures, e.g., Agrawal and Chase (2017a); V. Goyal, Pandey, Sahai, and Waters (2006a); Riepel and Wee (2022a).

Registered Functional Encryption (RFE), e.g., Branco et al. (2025); Chu, Lin, Qian, and Chen (2024); Datta and Pal (2023); Datta, Pal, and Yamada (2025); Francati et al. (2023); Zhu, Li, Zhang, Gong, and Qian (2024) extends the registration paradigm to FE introduced in Boneh, Sahai, and Waters (2011); O'Neill (2010), where users register public keys associated with specific functions, where upon successful decryption, recipients of encrypted data learn only the function's output on plaintext data. Like RBE and RABE, RFE enables users to self-generate key pairs, register only public information, and enables the KC to periodically update helper keys for users as public parameters evolve. RFE thus encapsulates both RBE and RABE within a unified, decentralized framework for secure computation over encrypted data as both RBE and RABE can be treated as special cases of RFE.

These constructions collectively define the RE framework, applicable to a wide range of cryptographic primitives in which trust is shifted from a fully trusted PKG to a semi-trusted, auditable KC. To ensure practical deployment, RE schemes aim to satisfy two key design goals:

- **Transparency and Auditability**: All KC operations must be deterministic and publicly verifiable, enabling full auditability and limiting the KC to an honest-but-curious role.

- **Compactness and Efficiency**: The size of public parameters and helper keys must grow at most polylogarithmically with the number of users, and the number of helper key updates per user should be minimal to support scalability.

By preserving the flexibility of advanced encryption primitives while mitigating the systemic vulnerabilities of centralized trust, the RE paradigm marks a significant evolution in the design of practical, secure access control systems.

Our Contributions. This article presents the first comprehensive survey of RE schemes. Our main contributions are as follows:

- *Taxonomy and Assessment Criteria*: We introduce a unified taxonomy that categorizes RE into three major branches—RBE, RABE, and RFE—and establish systematic assessment criteria spanning security, efficiency, and scalability.
- *Systematic Analysis*: We formally define the syntax, adversarial models, and security goals of RE schemes. We conduct a detailed analysis of representative constructions in each category, examining their design strategies, underlying cryptographic assumptions, update mechanisms, and distinguishing features.
- *Comparative Evaluation*: We compile comparative tables that highlight trade-offs across existing schemes, including parameter sizes, update costs, group assumptions, and adversarial robustness.
- *Design Insights*: Based on the analysis of RE designs, we provide design insights for developing efficient and effective RE schemes in terms of specific objectives.
- *Research Roadmap*: Drawing from our comprehensive analysis, we outline key open challenges and propose future research directions aimed at advancing the practicality, robustness, and functionality of RE systems.

Chapter Organization. The remainder of this paper is organized to provide a comprehensive examination of RE schemes. Section 3.2 introduces the background and preliminaries. Section 3.3 establishes a comprehensive taxonomy and assessment criteria for RE schemes. Sections 3.4, 3.5, and 3.6 present a systematic analysis of representative constructions for RBE, RABE, and RFE, respectively. Section 3.7 offers a unified analysis of the existing RE schemes and provide key design principles for developing

these RE schemes. Section 3.8 identifies future research directions within the field of RE. Finally, the concluding section 3.9 summarizes the key findings of this paper.

3.2 Background and Preliminaries

To facilitate a comprehensive understanding of the RE paradigm, this section introduces foundational concepts in identity-based cryptography and outlines the motivations that led to RE's development. We first define the core cryptographic roles and assumptions that underpin identity-based systems. We further examine the historical evolution from IBE to REs and conclude with a comparison to related paradigms that aim to reduce centralized trust.

3.2.1 Definitions and Trust Assumptions.

Definition 1 (Private Key Generator (PKG))**.** *A PKG is a trusted central authority in a public-key encryption system that maintains the master secret key and is responsible for generating private keys for all users in the system based on their identities/attributes/functions. Formally, a PKG is characterized by the following components:*

- ***Setup****: The PKG executes a probabilistic polynomial-time setup algorithm that takes as input a security parameter and outputs the system's public parameters and a master secret key. The public parameters are made available to all participants in the system.*

- ***Key Generation****: Upon receiving a key generation request from a user, the PKG executes a probabilistic polynomial-time private key generation algorithm that takes as input the master secret key and the user's identity/attributes or the authorized function, and outputs a corresponding private key.*

– ***Authentication Mechanism****: The PKG implements an authentication protocol to verify the legitimacy of each key generation request, ensuring that they key generation is only granted for an authorized entity who can prove their ownership of the requested identity/attributes/authorized function.*

Definition 2 (Key Curator (KC))**.** *A KC is a semi-trusted central authority in a public-key encryption system. The notion "semi-trusted" indicates that the KC is curious but honest: it honestly performs its algorithms but is curious about user information. The KC's primary responsibility is to register user public keys with their associated identities, attributes, or authorized functions. Formally, a KC is characterized by the following components:*

– ***Setup****: The KC executes a probabilistic polynomial-time setup algorithm that takes a security parameter as input. This algorithm outputs the system's public parameters, which are then made available to all participants in the system. This setup process is typically a one-time, foundational trusted operation.*

– ***Registration****: Upon receiving a user's public key along with their corresponding identity, attributes, or authorized function, the KC executes a deterministic polynomial-time user registration algorithm. This algorithm takes the public parameters and the user's information as input, and publicly updates the public parameters by associating the user's public key with their provided identity, attributes, or authorized function.*

– ***Update****: The KC runs a deterministic polynomial-time algorithm that takes as input the updated public parameters and a user's identity/attributes/authorized function. This algorithm then outputs a user helper key, which is used for specific system operations.*

– ***Authentication Mechanism****: The KC implements robust authentication protocols to verify the legitimacy of registration requests. This ensures that the KC only registers authorized entities who can successfully prove ownership of the requested identity, attributes, or authorized function.*

Definition 3 (Key Escrow Problem)**.** *The Key Escrow Problem refers to a fundamental security vulnerability in cryptographic systems where a trusted third party (escrow agent) possesses the capability to access, generate, or reconstruct the private cryptographic keys of system users, thereby enabling unauthorized decryption of confidential communications without the explicit consent or knowledge of the legitimate key holders. This problem was first identified by Shamir Shamir (1985) in the context of Identity-Based Encryption, where he noted that "the user's privacy is not protected against his own government or against an authority which issues the public directories."*

The system exhibits the key escrow problem if there exists a polynomial-time algorithm Recover *such that, for any user identity ID and corresponding key pair* $(pk_{ID}, sk_{ID}) \leftarrow \mathsf{KeyGen}(ID)$*, and for any ciphertext* $ct \leftarrow \mathsf{Encrypt}(pk_{ID}, m)$ *of a message* m*, the trusted authority* $\mathcal{A}$ *can compute a secret key* $sk'_{ID} \leftarrow \mathsf{Recover}(\mathcal{A}.state, ID)$ *that decrypts the ciphertext correctly, i.e.,* $\mathsf{Decrypt}(sk'_{ID}, ct) = m$ *with overwhelming probability, without requiring interaction with or authorization from the legitimate user holding* sk_{ID}*.*

The key escrow problem is particularly acute in Identity-Based Encryption (IBE) systems, where the Key Generation Center (KGC) can decrypt all ciphertexts and access all messages since it inherently possesses the capability to generate all user private keys. As Shamir originally observed Shamir (1985), this creates an inherent tension between the system's operational requirements and user privacy, since the trusted authority must

be capable of generating any user's private key to enable the identity-based paradigm to function.

Remark 1 (Historical Context)**.** *The key escrow problem was first identified by Shamir (1985) introducing the concept of Identity-Based Encryption. The problem became more prominent with the development of practical IBE schemes, particularly the Boneh and Franklin (2001)'s construction. Subsequent research has focused on various approaches to mitigate or eliminate this problem, including Accountable Authority IBE schemes, like V. Goyal, Lu, Sahai, and Waters (2008), and more recently, Registration-Based Encryption like S. Garg et al. (2018).*

Remark 2 (Security Implications)**.** *The key escrow problem presents several critical security concerns:*

- ***Privacy Violation:*** *Authorized third parties may gain access to encrypted data under certain circumstances, potentially compromising user privacy.*
- ***Single Point of Failure:*** *The escrow agent becomes a high-value target for adversaries seeking to compromise multiple users simultaneously.*
- ***Trust Concentration:*** *Users must place absolute trust in the escrow agent's integrity, security practices, and resistance to coercion.*
- ***Insider Threats:*** *Malicious or compromised personnel within the escrow organization can abuse their privileged access.*

Remark 3 (Distinction from Key Recovery)**.** *While related, key escrow differs from legitimate key recovery mechanisms. Key recovery systems provide access to plaintext outside of normal encryption/decryption channels, but the key escrow problem specifically refers to scenarios where this capability exists without explicit user consent or where it creates an inherent vulnerability in the system's security model.*

3.2.2 Historical Evolution.

3.2.2.1 From IBE to RBE. The development of RBE represents a revolutionary solution to cryptography's most persistent paradox: how to achieve the operational elegance of identity-based systems without surrendering user privacy to central authorities. This four-decade journey from Shamir's visionary concept to RBE's breakthrough illustrates both the allure of identity-based cryptography and the fundamental challenge that nearly rendered it impractical.

When Shamir (1985) introduced Identity-Based Encryption, he articulated a transformative vision: cryptographic systems where "arbitrary strings such as email addresses" could serve as public keys, eliminating the complexity of traditional certificate management. Yet Shamir himself recognized the Achilles' heel of his proposal, explicitly warning that "the user's privacy is not protected against his own government or against an authority which issues the public directories." This prescient observation established what would become known as the key escrow problem—the inescapable reality that any trusted PKG capable of issuing private keys could decrypt any message in the system.

For nearly two decades, IBE remained a theoretical curiosity until Boneh and Franklin (2001)'s groundbreaking work provided the first practical implementation using bilinear pairings. This achievement transformed IBE from mathematical abstraction to deployable technology, but it also made the key escrow problem urgently concrete. The promise of simplified key management came at the unacceptable cost of total privacy surrender to the PKG.

Subsequent decades witnessed a parade of increasingly sophisticated attempts to resolve the key escrow issue. Accountable Authority IBE schemes V. Goyal et al. (2008) added detection mechanisms for PKG misbehavior, certificate-based encryption sought to marry PKI robustness with IBE convenience, and various escrow-free IBE variants

emerged. Each approach demanded significant compromises: additional trusted parties, computational overhead, or byzantine key management procedures that negated the very simplicity that made IBE attractive.

The deadlock persisted until 2018, when S. Garg et al. (2018) achieved what many considered impossible with Registration-Based Encryption. Their breakthrough was architectural rather than algorithmic: instead of trying to constrain a powerful PKG, they eliminated its power entirely. In RBE, users "sample their own public and secret keys," maintaining complete control over their cryptographic materials. The KC serves only to "aggregate the public keys of all registered users and update the short public parameter," possessing "no knowledge of any secret key." Most remarkably, encryption remains genuinely identity-based, performed "using the recipient's identity and public parameters released subsequent to registration."

3.2.2.2 Toward Registered Encryption Cryptography.

RBE's success has catalyzed a fundamental paradigm shift in cryptographic design. The core insight—that public key aggregation enables identity-based operations without private key compromise—reveals the registration model's broader revolutionary potential beyond simple identity-based encryption.

This recognition immediately spawned RABE, which extends RBE's architectural principles to fine-grained access control. Users register not merely identities but complete attribute sets, enabling sophisticated policy-based encryption while preserving the escrow-free guarantee. The key curator aggregates attribute-based public keys without accessing private keys or sensitive attribute information, finally achieving the long-sought goal of policy-based encryption without centralized trust.

The logical culmination emerged in RFE, representing the most general realization of the registration paradigm. This framework enables users to register keys corresponding

to specific computational functions, supporting encrypted computation while maintaining both privacy and escrow-freedom. Authorized parties can compute designated functions over encrypted data without any central authority holding master keys capable of compromising the entire system.

These developments illuminate RBE's profound significance: it not only resolved the critical key escrow problem but also established a new foundational principle for cryptographic architecture. The registration approach, which clearly separates key generation from public parameter management, has since become the definitive template for escrow-free cryptographic systems across a broad spectrum of functionalities. Consequently, in this paper, we identify this paradigm as *registered encryption (RE)*.

This evolution from Shamir's original IBE through decades of partial solutions to the registration-based breakthrough represents more than incremental progress. It exemplifies cryptography's ultimate triumph: achieving Shamir's vision of practical identity-based cryptography while definitively resolving the privacy-functionality tension that had seemed irreconcilable. The progression from simple identity-based encryption through registration-based variants to registered functional encryption demonstrates not just technical advancement but a fundamental paradigm shift toward user-controlled cryptographic systems that preserve operational elegance while ensuring authentic privacy protection. After four decades of compromise, cryptography had finally achieved its Holy Grail: the full benefits of identity-based systems without sacrificing user autonomy to central authorities.

3.2.3 Comparison with Related Paradigms. RBE shares conceptual goals with several established paradigms against key-escrow issues while distinguishing itself through unique trust assumptions and architectural design choices. This section examines the key relationships and distinctions between RBE and related approaches.

3.2.3.1 Decentralized PKI (DPKI).

Decentralized PKI (DPKI) aims to establish fully decentralized systems for managing and verifying public keys and identities by replacing traditional Certificate Authorities with distributed ledgers such as blockchain networks. In DPKI systems, users manage their own cryptographic keys while registering public keys on immutable ledgers, thereby distributing trust across network participants through consensus mechanisms.

Several implementations exemplify this approach. Early proposals such as Certcoin introduced by Fromknecht, Velicanu, and Yakoubov (2014) leveraged Namecoin's distributed ledger, while more recent projects utilize Ethereum smart contracts, including Trustful presented by Dua, Barpanda, Kumar, and Tanwar (2020) and ETHERST demonstrated by Koa, Heng, and Chin (2025). Specialized applications have emerged for specific domains, such as D2XChain designed by Akram and Anaissi (2024) for drone services and lightweight solutions tailored to IoT ecosystems introduced by El-Hajj and Beune (2024). Corporate contexts are also exploring DPKI advantages, like Springer and Haindl (2024), with industry initiatives like 1Kosmos DPKI and broader Trustchain efforts in Hobson, France, Greenbury, Hare, and Wochner (2023) aligning with Self-Sovereign Identity (SSI) and Decentralized Identifiers (DIDs) principles for verifiable credentials. The WebOfTrustInfo DPKI initiative introduced by Christopher et al. (2017) further demonstrates ongoing development in this space.

Comparison with RBE: While both paradigms address key management challenges, they differ fundamentally in their architectural approaches and trust models. DPKI achieves complete decentralization by eliminating central points of control and relying on distributed consensus for integrity, as demonstrated by projects like DNS-On-Chain in Falzone (2023) that store and retrieve certificates on the Ethereum blockchain. In

contrast, RBE maintains a central KC for registration and system parameter distribution, though this curator notably does not hold any secret keys, thereby preventing key escrow.

The encryption processes also differ significantly. DPKI typically requires senders to retrieve recipients' individual public keys from decentralized registries, similar to certificate retrieval mechanisms. RBE streamlines this process by enabling senders to encrypt messages using only the recipient's identity and compact public parameters provided by the KC, substantially simplifying sender-side key management.

In essence, while DPKI fundamentally redistributes trust through complete decentralization, RBE focuses on efficient key management within a partially centralized framework that explicitly avoids the key escrow problem inherent in traditional identity-based systems.

3.2.3.2 Accountable Authority IBE (A-IBE).

Accountable Authority IBE (A-IBE) represents a sophisticated refinement of traditional Identity-Based Encryption that addresses the inherent key escrow problem through accountability mechanisms. While the PKG retains the ability to generate private keys, A-IBE schemes incorporate mechanisms to trace or prove malicious key generation or leakage by the PKG, thereby deterring misbehavior through transparency and auditability.

The theoretical foundations of A-IBE were established through early work on black-box traceability by V. Goyal et al. (2008), subsequently enhanced with improvements in ciphertext and private key efficiency by Libert and Vergnaud (2009). Research has since focused on strengthening accountability and traceability mechanisms, e.g., Kiayias and Tang (2015); Lai, Deng, Zhao, and Weng (2013a); Lee, Kim, and Lee (2023); Lee et al. (2023); L. Zhang, Yang, Song, and Wu (2023); Z. Zhao, Guo, et al. (2020); Z. Zhao, Wu, et al. (2020), developing more comprehensive security models, e.g., Sahai and Seyalioglu (2011); Singh, Acharya, and Dutta (2023); Tallón-Ballesteros

(2022), and adapting these concepts to specialized environments including cloud-assisted e-health systems like L. Yang, Liu, and Li (2022) and mobile e-commerce platforms like Han et al. (2016).

Comparison with RBE: The fundamental distinction between A-IBE and RBE lies in their respective approaches to resolving the key escrow problem and the operational role of central entities. A-IBE addresses key escrow through deterrence mechanisms, incorporating accountability and traceability for PKG actions, as evidenced by the "Dishonest PKG" security models that form the cornerstone of A-IBE design like Sahai and Seyalioglu (2011). RBE eliminates key escrow entirely by ensuring the central key curator never possesses secret keys, fundamentally redistributing key generation responsibility to individual users.

The key generation and distribution processes further differentiate these approaches. A-IBE maintains the traditional IBE model where users obtain private keys from the PKG through interactive protocols designed to enable accountability like V. Goyal et al. (2008). Conversely, RBE empowers users to independently generate their own public/secret key pairs, with the key curator's role limited to aggregating public keys into a compact master public key for encryption purposes rather than issuing secret keys.

These architectural differences fundamentally impact the trust models: A-IBE relies on the PKG's demonstrable honesty and the detectability of misbehavior, while RBE completely decentralizes secret key generation, removing the possibility of central authority compromise.

In summary, A-IBE preserves IBE's computational efficiency while ensuring PKG transparency and accountability. RBE redesigns the central authority's role to be entirely non-custodial, fundamentally eliminating key escrow possibilities by distributing the core key generation process among users themselves.

3.3 Taxonomy and Assessment Criteria

3.3.1 Taxonomy of RE. To systematically classify RE schemes, we propose a taxonomy based on two key dimensions: *functionality* and *construction methodology*.

This taxonomy provides a structured framework for analyzing and comparing RE schemes based on their functionality, construction techniques, and security foundations. It facilitates a deeper understanding of the landscape of RE research and highlights directions for future advancements, particularly in post-quantum security and functional enhancements.

By functionality. Based on the cryptographic functionality and access structures they support, RE schemes can be categorized into four main types:

- **Registration-Based Encryption (RBE):** Extends traditional Identity-Based Encryption (IBE) with a KC.
- **Registered Attribute-Based Encryption (RABE):** Incorporates attribute-based access control with a KC.
- **Registered Functional Encryption (RFE):** Employs the KC in Functional Encryptions, where users can learn specific functions of the encrypted plaintext instead of obtaining full decryption.

By construction. RE schemes can also be categorized based on their construction approaches, built upon underlying cryptographic primitives.

- **Non-Black-Box RE:** These schemes require deep modifications to cryptographic constructions, often integrating new hardness assumptions or novel techniques. Non-black-box cryptographic schemes lead to increased complexity and brittleness of their design, harder security proofs, and less robustness of future developments.

– **Black-Box RE:** These schemes leverage existing cryptographic primitives in a modular manner, ensuring compatibility with well-established encryption techniques.

3.3.2 Assessment Criteria of RE. For a systematic comparison of existing RE schemes, we present the assessment criteria with respect to security and performance. It is noted that the assessment criteria are proposed for fairly evaluating the properties claimed in different RE schemes.

3.3.2.1 Security Assessment Criteria.

– Security Model. If generic groups are involved in RE scheme's security proofs, it is considered that the scheme has security in generic group models (GGMs). In addition, according to whether random oracles are used in the security analysis, the security models are categorized into the standard model (STM) and the random oracle model (ROM). In the concrete security analysis, non-generic groups are better than generic groups, and STM are preferable to ROM. It is noted that the generic group model and the ROM are not comparable:

  * The GGM means that no property other than equality of gorup elements can be directly tested by adversaries. In other words, the adversaries cannot perform most of the group operations themselves and have to rely on queries to obtain the operation results. The GGM focus on the generic properties of mathenmatical groups, while non-GGM are based on well-studied hard problems, such as Decision Linear (DLIN) assumption, the k-Linear (k-LIN) assumption, the Decisional Bilinear Diffie-Hellman (DBDH) assumption, and so on.

    - The ROM means that random oracles are involved in the model. A random oracle is a black box that responds to each query by giving a random value chosen uniformly from its output domain. If a query is repeated, it returns the same value as before.
    - The GGM and the ROM can be utilized to proof secure simultaneously.

- Complexity Assumption. A RE scheme's security is usually reduced to the adopted complexity assumptions. It is more desirable to prove the security under the recognized assumptions, of which the form is concise and the complexity is proved. The security proofs under complexity assumptions of concise forms are technically challenging, because fewer parameters are provided by the assumption instance and used by the challengers.

3.3.2.2 Performance Assessment Criteria.

- *Size of Common Random String.* This parameter directly influences storage overhead and is determined by the number of registered users, functions, and attributes.

- *Size of Public Parameters.* This parameter affects both communication and storage overhead. Similar to the common random string, it scales with the number of registered users, functions, and attributes.

- *Size of Ciphertext.* The ciphertext size impacts communication and storage efficiency. It typically increases with the complexity of the associated access policy, the function's computational requirements, or the number of intended recipients.

- *Size of Helper Key.* This parameter contributes to both communication and storage overhead and is constrained by the number of registered users.

- *Number of Helper Key Updates.* This parameter indicates how frequently a user must update its helper key over its lifetime. The update frequency is influenced by the number of registered users.

- *Computation Cost.* The overall computation cost in RE comprises various computational complexities, including Setup Computation Cost, Key Generation Cost, Registration Computation Cost, Update Computation Cost, Encryption Computation Cost, and Decryption Computation Cost.

- *Group.* In black-box RE constructions, the underlying mathematical groups are categorized into prime-order and composite-order groups based on their order properties. Prime-order RE constructions are generally preferred over composite-order counterparts due to their efficiency. However, achieving full security in prime-order ABE is technically more challenging, as security proofs in such settings typically rely on composite-order groups.

- *User Universe.* The user universe in RE is classified into two categories: bounded and unbounded. A bounded user universe implies that the RE scheme is initialized with a fixed number of users, whereas an unbounded user universe allows for an indefinite number of users without a predefined limit during the setup phase.

3.4 Registration-Based Encryption (RBE)

Registration-Based Encryption (RBE) eliminates the need for a Private Key Generator (PKG) in traditional Identity-Based Encryption (IBE). It involves four key entities: the Cloud Service Provider (CSP), the Key Curator (KC), the Data Owner (DO), and the Data User (DU).

RBE begins with a one-time trusted setup to generate a common random string, which will be used for key generation, registration and encryption. The DU independently

generates a public-secret key pair, similar to traditional public-key encryption, and registers their public key along with their identity in cooperation with the KC. During this process, the KC updates the public parameters of the scheme.

Similar to IBE, when a DO shares data with a DU, the DO encrypts the data using the common random string, the public parameters and the DU's identity. To decrypt the data, the DU uses their secret key and a helper key, both of which correspond to the identity associated with the encrypted data.

Since the public parameters of the RBE scheme are updated whenever new DUs join the system, DUs must periodically refresh their helper keys over time. Notably, helper keys for each user can be computed publicly, and importantly, in RBE, the KC does not possess any secret information, ensuring a more transparent and trust-minimized system.

3.4.1 Syntax. Definition (Syntax of RBE). A registered identity-based encryption (RBE) scheme consists of PPT algorithms (Gen, Reg,Enc, Upd, Dec) working as follows. The Reg and Upd algorithms are performed by the KC.

- Setup. $Setup(\lambda) \rightarrow crs$. The probabilistic algorithm takes the security parameter λ as inputs and outputs a common random string crs. Some of the subroutines below will need a common random string crs, which could be sampled publicly using some public randomness beacon.

- Key generation. $KGen(\lambda) \rightarrow (pk, sk)$. The probabilistic algorithm KGen takes as input the security parameter λ and outputs a pair of public/secret keys (pk, sk). Note that these are only public and secret keys, not the encryption or decryption keys. The key generation algorithm is run by any honest DU locally who wants to register itself into the system.

- Registration. $Reg^{[aux]}(crs, pp, id, pk) \rightarrow pp'$. The deterministic algorithm Reg takes as input the common random string crs, current public parameters pp, a registering DU's identity id and a public key pk (supposedly for the identity id), and it outputs pp' as the updated public parameters.

- probabilistic. $Enc(crs, pp, id, m) \rightarrow ct$. The randomized algorithm Enc takes as input the common random string crs, the current public parameters pp, a DU's identity id and a plaintext message m and outputs a ciphertext ct.

- Update. $Upd^{[aux]}(pp, id) \rightarrow hpk$. The deterministic algorithm Upd takes as input the current public parameters pp and an identity id, and generates an update helper key hpk that can help DU id to decrypt messages.

- Decryption. $Dec(sk, hpk, ct) \rightarrow m$: The deterministic decryption algorithm Dec takes as input a secret key sk, an helper key hpk, and a ciphertext ct, and it outputs a message m.

Definition (Compactness and Efficiency of RBE).

- **Compactness** The public parameters and helper keys must be short, i.e., $\mathcal{P}(\lambda, \log n)$, where n is the number of registered users and λ is the security parameter.

- **Efficiency of runtime:** The registration process and the helper key updating must be efficient, i.e., they must run in time $\mathcal{P}(\lambda, \log n)$ per user registration.

- **Efficiency of updates:** The number of times that a DU must update helper keys must be low, i.e., $\mathcal{P}(\log n)$ over the lifetime of the system.

3.4.2 Adversarial Model and Security Goals.

In RBE, all KCs are considered semi-trusted and transparent. Semi-trusted stands for that the KC honestly follows the protocol but may attempt to learn sensitive user information, such as secret keys and shared data. Transparent stands for that the KC operates deterministically and retains no secrets, making it fully auditable by any party.

A secure RBE system must satisfy the following security properties:

- Data Confidentiality: Unauthorized DUs must be unable to decrypt ciphertexts. Additionally, the KC should not have the capability to decrypt ciphertexts without authorization.

- Collusion Resistance: a RBE system should prevent collusion attacks from unauthorized users and the KC.

3.4.3 Research Status.

S. Garg et al. (2018) first introduced the notion of registration-based encryption (RBE) by eliminating the PKG from identity-based encryption (IBE). They proposed two RBE schemes: one based on somewhere statistically binding hash functions in Hubacek and Wichs (2015) and indistinguishability obfuscation (IO) like Barak et al. (2001); S. Garg, Gentry, et al. (2016) and another relying on hash-garbling schemes like Brakerski, Lombardi, Segev, and Vaikuntanathan (2018); Cho et al. (2017); Döttling and Garg (2017b); Döttling, Garg, Hajiabadi, and Masny (2018) (built from standard assumptions such as CDH, Factoring, or Learning With Errors (LWE) like Peikert (2009); Regev (2009)). However, their constructions lacked efficiency, leaving open the challenge of designing an efficient RBE scheme based on standard assumptions.

To address this gap, S. Garg et al. (2019a) later proposed an efficient RBE scheme based on standard assumptions. Their construction leverages a Red-Black Merkle Tree

and hash-garbling primitives. Additionally, they introduced anonymous RBE, where the encrypted identity remains hidden alongside the message, realized using blind public-key encryption (PKE), e.g., Bellare, Boldyreva, Desai, and Pointcheval (2001); Mohassel (2010), and blind hash-garbling schemes.

To further improve efficiency, Cong et al. (2021) optimized ciphertext size by replacing Merkle trees in RBE with crit-bit trees Langley (2008) (a variant of PATRICIA tries Morrison (1968)) and increasing the size of RBE's public parameters. Their approach reduced ciphertext size by 57.5% and decryption computation cost by 30%.

Mahmoody et al. (2022) established a provable lower bound for update RBE, proving a trade-off between the number of updates and the length of public parameters in any scheme with a fixed number of updates.

All the above schemes rely on non-black-box cryptographic techniques, making them highly impractical. To overcome this limitation, Glaeser et al. (2023a) were the first to propose an RBE scheme using only black-box cryptographic primitives from standard assumptions, based on bilinear maps. They also implemented a prototype of their RBE scheme, marking the first-ever implementation. Further, Hajiabadi et al. (2023) formally proved that black-box constructions of RBE are impossible in both random trapdoor permutations (TDPs) models and the generic group model.

Moreover, none of the previously mentioned schemes support an unbounded number of users, making them unsuitable for large-scale applications. Döttling et al. (2023) introduced laconic cryptography from LWE, enabling RBE schemes to support an unbounded number of users with public parameters that scale logarithmically with the number of users. Fiore et al. (2023) further extended this concept by proposing a black-box RBE scheme that supports an unbounded user universe, leveraging cuckoo hashing Pagh and Rodler (2004) in cryptographic constructions.

However, all the discussed schemes assume an honest KC, which may not be realistic in real-world scenarios. To address this, Goyal and Vusirikala R. Goyal and Vusirikala (2020) introduced verifiable RBE (VRBE), which ensures security even with a malicious KC. In VRBE, users can obtain short proofs from the KC verifying correct (and unique) registrations as well as proofs of non-registration for unregistered identities. Their scheme is constructed using CDH, Factoring, and LWE assumptions. Further securing RBE without a trusted setup, Dujmovic, Malavolta, and Qi (2025) extends the foundational work of S. Garg et al. (2018) and replaces the trusted setup with an honest user assumption. It utilizes non-interactive witness indistinguishable (NIWI) proofs secure against chosen-statement attacks, along with a re-randomizable RBE design. A key tradeoff, however, is that users are required to update their keys following every new registration, which limits scalability in dynamic environments.

Furthermore, Dujmovic et al. (2025) proposed an RBE in plain model. Their construction extends S. Garg et al. (2018)'s work and introduces an honest user instead of a trusted setup based on non-interactive witness indistinguishable (NIWI) proofs secure against chosen statements attack and re-randomizable RBE. However, it requires that users must be updated upon every new registration.

More recently, Q. Wang et al. (2023) proposed a blockchain-based Nakamoto (2008) RBE framework, introducing transparency and decentralization by utilizing smart contracts Wood et al. (2014). This approach eliminates centralized key management, shifting the responsibility from a KC to individual participants, while keeping publicly upgradable parameters on-chain.

3.4.4 Comparison.

Table 1. Comparisons of RBE schemes.

Schemes	**Parameter Size**					
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	$\#(Upd)$
GHMR18-1 S. Garg et al. (2018)	-	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
GHMR18-2 S. Garg et al. (2018)	-	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
GHMR19 S. Garg et al. (2019a)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
CES21 Cong et al. (2021)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
MQR22 Mahmoody et al. (2022)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\frac{\log n}{\log \log n})$
GKMR23 Glaeser et al. (2023a)	$\mathcal{P}(\lambda, \sqrt{n})$	$\mathcal{P}(\sqrt{n})$	$\mathcal{P}(\sqrt{n})$	1	$\mathcal{P}(\sqrt{n})$	$\mathcal{P}(\sqrt{n})$
DKLL23 Döttling et al. (2023)	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(n)$
FKP23 Fiore et al. (2023)	$\mathcal{P}(\sqrt{\lambda n}, \sqrt{n})$	$\mathcal{P}(n\sqrt{n})$	$\mathcal{P}(\sqrt{n})$	$\mathcal{P}(\sqrt{n})$	$\mathcal{P}(\sqrt{n})$	$\mathcal{P}(\sqrt{n})$
GV20 R. Goyal and Vusirikala (2020)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
WLWG23 Q. Wang et al. (2023)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
DMQ25 Dujmovic et al. (2025)	-	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log n) + \mathcal{P}(n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(n)$

Schemes	**Computation Cost**					
	Setup	KeyGen	Reg	Update	Enc	Dec
GHMR18-1 S. Garg et al. (2018)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
GHMR18-2 S. Garg et al. (2018)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$
GHMR19 S. Garg et al. (2019a)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$
CES21 Cong et al. (2021)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$
MQR22 Mahmoody et al. (2022)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$
GKMR23 Glaeser et al. (2023a)	$\mathcal{P}(\lambda, \sqrt{n})$	$\mathcal{P}(\lambda, \sqrt{n})$	$\mathcal{P}(\sqrt{n})$	1	1	1
DKLL23 Döttling et al. (2023)	$\mathcal{P}(\lambda, \log n)$	1	$\mathcal{P}(n)$	1	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$

Continued on next page

Table 1. Comparisons of RBE schemes (Continued)

Schemes	**Parameter Size**					$\#(Upd)$
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	
FKP23 Fiore et al. (2023)	$\mathcal{P}(\lambda, \sqrt{n})$	$\mathcal{P}(\lambda, \sqrt{n})$	$\mathcal{P}(\sqrt{n})$	1	$\mathcal{P}(\sqrt{n})$	1
GV20 R. Goyal and Vusirikala (2020)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log^2 n)$	$\mathcal{P}(\log^2 n)$
WLWG23 Q. Wang et al. (2023)	$\mathcal{P}(\lambda)$	1	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
DMQ25 Dujmovic et al. (2025)	-	$\mathcal{P}(\lambda, n, \log n)$	$\mathcal{P}(n, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$

Schemes	**black-box**	**Universe**	**Group**	**Security**	**Adv**	**Assump**
GHMR18-1 S. Garg et al. (2018)	○	B	-	STM	Std	IO, SSBH
GHMR18-2 S. Garg et al. (2018)	○	B	-	STM	Std	HG
GHMR19 S. Garg et al. (2019a)	○	B	-	STM	Std	HG
CES21 Cong et al. (2021)	○	B	-	STM	Std	HG
MQR22 Mahmoody et al. (2022)	○	B	-	STM	Std	-
GKMR23 Glaeser et al. (2023a)	●	B	Prime	STM	Std	DBDH, q-type
DKLL23 Döttling et al. (2023)	●	UnB	-	STM	Std	LWE
FKP23 Fiore et al. (2023)	●	UnB	-	STM	Std	LWE
GV20 R. Goyal and Vusirikala (2020)	○	B	-	STM	MKC	HG
WLWG23 Q. Wang et al. (2023)	○	B	-	STM	MKC	SC
DMQ25 Dujmovic et al. (2025)	○	B	-	STM	MKC	IO,NIWI

Notes:

$\mathcal{P}(\cdot)$ indicates a polynomial function; ○: non-black-box; ●: black-box.

B: bounded user universe; UnB: unbounded user universe; Std: standard adversary.

The comparison of RBE schemes in Table 1 highlights important trade-offs between the size of the common reference string $|crs|$, the size of the auxiliary

information $|aux$, the size of the public parameter $|pp|$, the size of the ciphertext $|ct|$, the size of the helper key $|hpk|$, the number of helper key updates $\#(Upd)$, the computation efficiencies of setup, key generation, registration, update, encryption, and decryption, black-box vs. non-black-box, user universe size, the type of group, the security model, the adversary type, and the underlying complexity assumptions.

Early RBE constructions, such as GHMR18-1 and GHMR18-2 introduced by S. Garg et al. (2018), GHMR19 introduced by S. Garg et al. (2019a), and CES21 introduced by Cong et al. (2021), primarily focus on achieving compact ciphertexts and public parameters that scale polylogarithmically in the number of users n, e.g., $\mathcal{P}(\log n)$. These schemes are among the first to achieve sub-linear parameter sizes compared to traditional IBE schemes and offer standard model security. However, a notable limitation of these constructions is their reliance on large auxiliary information aux of size $\mathcal{P}(\lambda, n)$, which may hinder practical deployment when n is large. Furthermore, their update complexity remains $\mathcal{P}(\log n)$, which, although sub-linear, still poses a challenge for systems requiring real-time revocations and updates.

A significant step forward is observed in recent RBE constructions such as GKMR23 introduced by Glaeser et al. (2023a), DKLL23 introduced by Döttling et al. (2023), and FKP23 introduced by Fiore et al. (2023), which focus on improving both parameter efficiency and update cost. Notably, GKMR23 introduced by Glaeser et al. (2023a) achieves constant-size ciphertexts (size 1), a major advancement over previous schemes, while also reducing update complexity to $\mathcal{P}(\sqrt{n})$. This scheme is also among the first black-box constructions that avoid relying on non-standard or circuit-specific techniques and is built in prime-order groups under the DBDH and q-type assumptions, providing strong security guarantees in standard models.

Similarly, FKP23 introduced by Fiore et al. (2023) achieves sub-linear parameter sizes (e.g., $\mathcal{P}(\sqrt{n})$ for most parameters) and constant-time updates, while supporting unbounded universes, meaning it does not require prior knowledge of all possible user identities. This is crucial for practical applications where user registration is dynamic and unpredictable. DKLL23 introduced by Döttling et al. (2023), based on LWE assumptions, also supports unbounded universes and achieves competitive parameter sizes, though some costs (such as auxiliary information) remain logarithmic in n.

Moreover, recent schemes like GV20 introduced by R. Goyal and Vusirikala (2020), WLWG23 introduced by Q. Wang et al. (2023), and DMQ25 introduced by Dujmovic et al. (2025) address malicious key generation attacks (by malicious KC), an essential feature for strengthening security in adversarial settings where registration authorities might be compromised. Among these, DMQ25 introduced by Dujmovic et al. (2025) presents a distinctive approach with a unique trade-off: it achieves a relatively large ciphertext size of $\mathcal{P}(\log n) + \mathcal{P}(n)$ but offers the highest number of updates at $\mathcal{P}(n)$, making it potentially suitable for scenarios requiring frequent revocation events. However, this comes at the cost of substantial computation overhead during key generation and registration, both scaling as $\mathcal{P}(n, \log n)$. DMQ25 also relies on indistinguishability obfuscation (IO) and non-interactive witness indistinguishable proofs (NIWI) assumptions, which are considered stronger cryptographic primitives than those used in some other schemes.

MQR22 introduced by Mahmoody et al. (2022) offers an interesting optimization in the update complexity, achieving $\mathcal{P}(\frac{\log n}{\log \log n})$ updates which improves upon the standard $\mathcal{P}(\log n)$ bound of earlier schemes. Meanwhile, WLWG23 introduced by Q. Wang et al. (2023) maintains parameter efficiency similar to earlier constructions but is notable

for incorporating smart contracts (SC) as its underlying assumption, representing an exploration of blockchain-based security for RBE systems.

In summary, while earlier schemes laid the groundwork for RBE with compact structures and standard security, more recent constructions such as GKMR23 introduced by Glaeser et al. (2023a), DKLL23 introduced by Döttling et al. (2023), and FKP23 introduced by Fiore et al. (2023) mark significant progress by reducing ciphertext and update sizes, supporting unbounded universes, and achieving black-box constructions under standard assumptions. Concurrently, schemes like DMQ25 introduced by Dujmovic et al. (2025) explore alternative trade-offs, prioritizing update frequency over parameter compactness. These diverse advancements collectively move RBE closer to practical deployment in large-scale systems requiring dynamic and efficient user management, with different schemes offering optimal solutions for varying application requirements and security priorities.

3.5 Registered Attribute-Based Encryption (RABE)

Similar to RBE, registered Attribute-Based Encryption (RABE) eliminates the need for a Private Key Generator (PKG) in traditional attribute-Based Encryption (ABE), particularly Ciphertext-Based ABE (CP-ABE). It also involves four key entities: the Cloud Service Provider (CSP), the Key Curator (KC), the Data Owner (DO), and the Data User (DU).

The cooperations between entities are the same as those of RBE except attributes and access policies involved as below:

- In RABE, the setup process generates the common random string associated with the size of the attribute universe.
- The registration process aggregates the DU's public key together with a set of DU's attributes, which is difference from RBE.

– The encryption process encrypts data associated with an access policy. Only the DU whose registered attribute set satisfies the associated access policy can decnrypt the data correctly.

3.5.1 Syntax. Definition (Access structures introduced by Beimel (1996a)). Let S be a set and let 2^S denote the power set of S (i.e., the set of all subsets of S). An access structure on S is a collection of non-empty subsets $\mathbb{A} \subseteq 2^S \setminus \varnothing$. We refer to the elements of $\mathbb{A}$ as the authorized sets and those not in $\mathbb{A}$ as the unauthorized sets. We say an access structure is monotone if for all sets $B, C \in s^S$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.

Definition (Linear Secret Sharing Scheme (LSSS) introduced by Beimel (1996a)). Let $\mathcal{P}$ be a set of DUs. A linear secdret sharing scheme over a ring $\mathbb{Z}_N$ for $\mathcal{P}$ is a pair $(\mathbf{M}, \rho)$, where $\mathbf{M} \in \mathbb{Z}_N^{l \times n}$ is a "share-generating" matrix and $\rho : [l] \rightarrow \mathcal{P}$ is a "row-labeling" function. The pair $(\mathbf{M}, \rho)$ satisfy the following properties:

– **Share generation:** To share a value $s \in \mathbb{Z}_N$, sample $v_2, \cdots, v_n \leftarrow_R \mathbb{Z}_N$ and define the vector $\mathbf{v} = [s, v_2, \cdots, v_n \leftarrow_R]^\top$. Then $\mathbf{u} = \mathbf{Mv}$ is the vector of shares where $u_i \in \mathbb{Z}_N$ belongs to party $\rho(i)$ for each $i \in [l]$.

– **Share reconstruction:** Let $S \subseteq \mathcal{P}$ be a set of parties and let $I_S = \{i \in [l] : \rho(i) \in S\}$ be the row indices associated with S. Let $\mathbf{M}_S \in \mathbb{Z}_N^{|I_S| \times n}$ be the matrix formed by taking the subset of rows in $\mathbf{M}$ that are indexed by I_S. If S is an authorized set of DUs, then there exists a vect $\boldsymbol{\omega}_S \in \mathbb{Z}_N^{|I_S|}$ such that $\boldsymbol{\omega}_S^\top \mathbf{M}_S = \mathbf{e}_1^\top$, where $\mathbf{e}_1^\top = [1, 0, \cdots, 0]$ denotes the first elementary basis vector. Conversely, if S is an unauthorized sests of parties, then $\mathbf{e}_1^\top$ is not in the row-span of $\mathbf{M}$ (i.e., there does not exist $\boldsymbol{\omega}_S \in \mathbb{Z}_N^{|S|}$ where $\boldsymbol{\omega}_S^\top \mathbf{M}_S = \mathbf{e}_1^\top$).

Definition (Syntax of RABE). A registered attribute-based encryption (RABE) scheme consists of PPT algorithms (Gen, Reg,Enc, Upd, Dec) working as follows. The Reg and Upd algorithms are performed by the KC.

- Setup. $Setup(\lambda, |\mathcal{U}|) \rightarrow crs$. The probabilistic algorithm takes the security parameter λ and the size of the attribute universe $|\mathcal{U}|$ as inputs and outputs a common random string crs. Some of the subroutines below will need a common random string crs, which could be sampled publicly using some public randomness beacon.

- Key generation. $KGen^{[aux]}(crs) \rightarrow (pk, sk)$. The probabilistic algorithm KGen takes as input the common random string crs and outputs a pair of public/secret keys (pk, sk). Note that these are only public and secret keys, not the encryption or decryption keys. The key generation algorithm is run by any honest DU locally who wants to register itself into the system.

- Registration. $Reg^{[aux]}(crs, pp, S, pk) \rightarrow pp'$. The deterministic algorithm Reg takes as input the common random string crs, current public parameters pp, a set of user attributes $S \subseteq \mathcal{U}$ and a public key pk, and it outputs pp' as the updated public parameters.

- Encryption. $Enc(crs, pp, \mathbb{A}, m) \rightarrow ct$. The probabilistic algorithm Enc takes as input the common random string crs, the public parameters pp, an access policy structure $\mathbb{A}$ and a plaintext message m and outputs a ciphertext ct.

- Update. $Upd^{[aux]}(pp, pk) \rightarrow hpk$. The deterministic algorithm Upd takes as input the current public parameters pp and a user public key pk, and outputs the corresponding helper key hpk that can help the user to decrypt its messages.

– Decryption. $Dec(sk, hpk, ct) \rightarrow m$: The deterministic decryption algorithm Dec takes as input a secret key sk, a helper key hpk, and a ciphertext ct, and it outputs a message m.

Definition (Compactness and Efficiency of RABE).

– **Compactness** The public parameters and helper keys must be short, i.e., $|pp_i| = \mathcal{P}(\lambda, |\mathcal{U}|, \log i)$ for all $i \in [n]$ and $|hpk| = \mathcal{P}(\lambda, |\mathcal{U}|, \log n)$, where λ is the security parameter, n is the number of registered users, and $|\mathcal{U}|$ is the size of attribute universe.

– **Efficiency of updates:** The number of times that a DU must update helper keys must be low, i.e., $\mathcal{P}(\log n)$ over the lifetime of the system.

3.5.2 Adversarial Model and Security Goals. Same as RBE, KCs in RABE are also considered semi-trusted and transparent. Besides, a secure RABE system must satisfy the following security properties:

– Data Confidentiality: DUs with unauthorized attribute sets must be unable to decrypt target ciphertexts. Additionally, the KC should not have the capability to decrypt ciphertexts without authorization.

– Collusion Resistance: Even if the adversary registers multiple keys and corrupts multiple users, it should not be able to combine attributes from different users to decrypt a ciphertext unless one of them already satisfies the policy.

3.5.3 Research Status. Hohenberger et al. (2023) first introduced Registered ABE (RABE), defining efficiency in terms of polylogarithmic scaling of public parameters, helper keys, ciphertexts, and encryption/decryption times relative to the number of registered users. Their scheme, which supports a bounded number of users

and policies, is constructed using black-box cryptography, including Linear Secret Sharing Scheme (LSSS)-based access structures like V. Goyal, Pandey, Sahai, and Waters (2006b); A. Lewko, Okamoto, Sahai, Takashima, and Waters (2010a); A. Lewko and Waters (2011) and composite-order pairing groups. Security is proven under subgroup decision assumptions in the standard model.

To improve RABE, Zhu et al. (2023) proposed a RABE scheme for arithmetic branching programs, based on the k-Lin assumption in prime-order bilinear groups. Additionally, they extended the framework to support span-program-based RABE and inner-product predicate RABE, following the blueprint established by Hohenberger et al. (2023).

However, a key limitation of the above schemes is that the size of the common random string (crs) scales quadratically with the number of users. To address this, Freitag et al. (2023) introduced a RABE scheme based on plain witness encryption like S. Garg, Gentry, Sahai, and Waters (2013) and function-binding hash functions under the LWE assumption, which supports an unbounded number of users. Their construction enables general policies over a super-polynomial attribute universe but achieves only selective security in the random oracle model. However, their construction is based on non-black-box use of cryptographic techniques.

Further optimizations were proposed by R. Garg et al. (2024), who introduced two approaches to reduce crs size in pairing-based RABE schemes. One approach utilizes a combinatorial technique based on progression-free sets, e.g., Behrend (1946); Elkin (2010); Erdös and Turán (1936); Moser (1950), leading to a sub-quadratic crs size in terms of the number of users. Another approach relies on a partitioning-based argument like Boneh and Boyen (2004); Waters (2011) for security analysis, resulting in a crs size independent of the attribute universe size. After that, Datta et al. (2025) achieves

compatible complexity to S. Garg et al. (2013)'s work except the crs. Specifically, the size of public parameters, ciphertext, and helper keys are independent on the number of users.

Beyond encryption, Y. Zhang et al. (2024) proposed Registered Attribute-Based Signatures (RABS), which differ from RABE in that policies and registered attributes are tied to user-generated signatures, allowing anyone to verify signatures using only public parameters. This approach provides fine-grained authentication control while ensuring anonymous message authentication. Their work presents the first formal definition of RABS and constructs the first fully secure RABS scheme over prime-order groups using the k-Lin assumption in the standard model. Both of these approaches achieve selective security.

To support large scale attributes and more expressive policies, Attrapadung and Tomida (2024) construct the first RABE scheme for monotone span programs with completely unbounded property. The completely unbounded property means no limitation on user attribute sets and policies associated with ciphertexts. Their construction is based on the pair encoding scheme (PES) like Agrawal and Chase (2017c); Attrapadung (2014); Attrapadung and Tomida (2020). Various PES transformation were demonstrated to support more expressive predicates. From another aspect, Champion, Hsieh, and Wu (2025) proposed a novel key-policy RABE based on LWE assumptions.

Recently, Zhu, Zhang, Chen, Gong, and Qian (2025) proposed the first black-box construction of RABE that supports general circuits. Their scheme achieves selective security under the evasive LWE assumption. Notably, it preserves desirable features from pairing-based RABE schemes—such as an unbounded user universe and transparent setup—as seen in prior works like Döttling et al. (2023); Fiore et al. (2023); Freitag et al. (2023). In contrast, existing RABE constructions for circuits under lattice assumptions

have so far relied on non-black-box techniques, typically leveraging witness encryption like Freitag et al. (2023). Following Zhu et al. (2025)'s work, Champion et al. (2025) proposed an RABE based on stronger security assumption, i.e., l-succinct LWE in random oracle model. Based on this construction, they introduced an adaptively-secure (distributed) broadcast encryption.

3.5.4 Comparison.

Table 2. Detailed Comparisons of RABE schemes.

Schemes	**Parameter Size**					$\#(Upd)$
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	
HLWW23 Hohenberger et al. (2023)	$n^2 \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$n^2 \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\mathcal{P}(\lambda, \|P\|, \log n)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\mathcal{P}(\log n)$
ZZGQ23 Zhu et al. (2023)	$n^2 \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|)$	-	$(\|sk\| + \|\mathcal{U}\|) \cdot \mathcal{P}(\lambda)$	$(\|sk\| + \|ct\|) \cdot \mathcal{P}(\lambda)$	$(\|sk\| + \|\mathcal{U}\|) \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$
FWW23 Freitag et al. (2023)	-	-	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\lambda, \|x\|)$	$\mathcal{P}(\lambda, \|x\|, \|m\|, \log n)$	$\mathcal{P}(\log n)$
ZZZG24 Y. Zhang et al. (2024)	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\lambda, \|P\|, \log^2 n)$	$\mathcal{P}(\lambda, \|P\|, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\lambda, \|x\|, \log n)$	$\mathcal{P}(\log n)$
GLWW24-1 R. Garg et al. (2024)	$\mathcal{P}(\lambda, n^{1+\mathcal{O}(1)})$	$\mathcal{P}(\lambda, \|\mathcal{U}\|, n)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|\mathcal{U}\|)$	$\mathcal{P}(\log n)$
GLWW24-2 R. Garg et al. (2024)	$\mathcal{P}(\lambda, n^2)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|, n)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|\mathcal{U}\|)$	$\mathcal{P}(\log n)$
DPY24-ABE Datta et al. (2025)	$n^2 \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	-	$\mathcal{P}(\lambda, \|\mathcal{U}\|)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|\mathcal{U}\|)$	$\mathcal{P}(\log n)$
AT24 Attrapadung and Tomida (2024)	$\mathcal{P}(\lambda, n^2, \|\mathcal{U}\|^2)$	-	$\|\mathcal{U}\|^2 \cdot \mathcal{P}(\lambda)$	$n\|\mathcal{U}\| \cdot \mathcal{P}(\lambda)$	$\|\mathcal{U}\|^2 \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$
ZZCG25 Zhu et al. (2025)	$\mathcal{P}(\lambda, k, l_c)$	-	$\mathcal{P}(\lambda, k, l_c, \log n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\log n)$
CHW25 Champion et al. (2025)	$(n^2 + \|\mathcal{U}\|^2) \cdot \mathcal{P}(\lambda, k)$	-	$\mathcal{P}(\lambda, k, \mathcal{L}(n))$	$\mathcal{P}(\lambda, k, \mathcal{L}(n))$	$\mathcal{P}(\lambda, k)$	$\mathcal{P}(\log n)$

Schemes	**Computation Cost**					
	Setup	KeyGen	Reg	Update	Enc	Dec

Continued on next page

Table 2. Detailed Comparisons of RABE schemes (Continued)

Schemes	**Parameter Size**					$\#(Upd)$
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	
HLWW23 Hohenberger et al. (2023)	$\mathcal{P}(\lambda, \|\mathcal{U}\|, n)$	$n \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$n \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(n)$	$\mathcal{P}(n)$
ZZGQ23 Zhu et al. (2023)	$\mathcal{P}(\lambda, \|\mathcal{U}\|, n)$	$\|sk\| \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\|sk\| \cdot \mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(n)$	$\mathcal{P}(n)$
FWW23 Freitag et al. (2023)	$\mathcal{P}(\lambda, \|x\|)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\lambda, \|x\|, \log n)$	$\mathcal{P}(\lambda, \|x\|, \|m\|, \log n)$	$\mathcal{P}(\lambda, \|x\|, \|m\|, \log n)$
ZZZG24 Y. Zhang et al. (2024)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\log n)$
GLWW24-1 R. Garg et al. (2024)	$\mathcal{P}(\lambda, \|\mathcal{U}\|)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\|\mathcal{U}\|, n)$	$\mathcal{P}(\|\mathcal{U}\|, \log n)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|P\|)$
GLWW24-2 R. Garg et al. (2024)	$\mathcal{P}(\lambda, \|\mathcal{U}\|)$	$\mathcal{P}(\lambda, \log n)$	$\mathcal{P}(\|\mathcal{U}\|, n)$	$\mathcal{P}(\|\mathcal{U}\|, \log n)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|P\|)$
DPY24-ABE Datta et al. (2025)	$\mathcal{P}(\lambda, \|\mathcal{U}\|, \log n)$	$\log^2 n \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\|\mathcal{U}\|, \log n)$	$\mathcal{P}(\|\mathcal{U}\|, \log n)$	$\mathcal{P}(\|P\|)$	$\mathcal{P}(\|P\|)$
AT24 Attrapadung and Tomida (2024)	$\mathcal{P}(\lambda, n^2, \|\mathcal{U}\|^2)$	$n \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, \|\mathcal{U}\|^2, n^2)$	$\mathcal{P}(\lambda, n, \|\mathcal{U}\|^2)$	$n\|\mathcal{U}\| \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n, \|\mathcal{U}\|^2)$
ZZCG25 Zhu et al. (2025)	$\mathcal{P}(\lambda, k, l_c)$	$k \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, k, l_c, n, \log n)$	$n \cdot \mathcal{P}(\log n)$	$\mathcal{P}(\lambda, k, l_c, n, \log n)$	$n \cdot \mathcal{P}(\log n)$
CHW25 Champion et al. (2025)	$(n^2 + \|\mathcal{U}\|^2) \cdot \mathcal{P}(\lambda, k, \mathcal{L}(n))$	$\mathcal{P}(\lambda, k, \mathcal{L}(n))$	$\mathcal{P}(\lambda, k, \mathcal{L}(n), n)$	$\mathcal{P}(\lambda, k, \mathcal{L}(n), n)$	$\mathcal{P}(\lambda, k, \mathcal{L}(n))$	$\mathcal{P}(\lambda, k, \mathcal{L}(n))$

Schemes	**black-box**	**Universe**	**Group**	**Policy**	**Security**	**Assump**
HLWW23 Hohenberger et al. (2023)	•	B	Composite	Span	STM	SDP, Factoring
ZZGQ23 Zhu et al. (2023)	•	B	Prime	ABP	STM	k-Lin
FWW23 Freitag et al. (2023)	○	UnB	Prime	General	ROM	LWE
ZZZG24 Y. Zhang et al. (2024)	•	B	-	ABP	STM	k-Lin
GLWW24-1 R. Garg et al. (2024)	•	B	Prime	Span	STM	q-type
GLWW24-2 R. Garg et al. (2024)	•	B	Prime	Span	STM	q-type
DPY24-ABE Datta et al. (2025)	•	B	Prime	Span	G-ROM	-

Continued on next page

Table 2. Detailed Comparisons of RABE schemes (Continued)

Schemes	Parameter Size					$\#(Upd)$
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	
AT24 Attrapadung and Tomida (2024)	•	UnB	Prime	General	STM	MDDH
ZZCG25 Zhu et al. (2025)	•	UnB	-	General	STM	LWE
CHW25 Champion et al. (2025)	•	B	-	K-General	ROM	l-LWE

Notes:

SDP: subgroup decision problem; Span: monotone span program; ABP: arithmetic branching program.

$|x|$: length of attribute string; $|P|$: length of access policy; $\mathcal{U}$: size of attribute universe.

$|m|$: length of message; $|sk|$ / $|ct|$: components of secret key / ciphertext.

k: depth of circuits; l_c: size of a circuit; l-LWE: l-succinct LWE.

$\mathcal{L}(n)$: parameter generation function for lattice; K-General: key-policy based RABE.

Table 2 presents a comparative analysis of existing RABE schemes based on various efficiency metrics, the size of the common reference string $|crs|$, the size of the auxiliary information $|aux$, the size of the public parameter $|pp|$, the size of the ciphertext $|ct|$, the size of the helper key $|hpk|$, the number of helper key updates $\#(Upd)$, the computation efficiencies of setup, key generation, registration, update, encryption, and decryption, black-box vs. non-black-box, user universe size, the type of group, the functionality of policy, the security model, and the underlying complexity assumptions. Based on this analysis, we highlight the most efficient schemes in key categories.

In terms of parameter sizes, FWW23 introduced by Freitag et al. (2023) and ZZZG24 introduced by Y. Zhang et al. (2024) achieve the smallest overall public parameters, ciphertexts, and auxiliary data, with sizes bounded by $\mathcal{P}(\lambda, \log n)$ and $\mathcal{P}(\lambda, |P|, \log n)$, making them highly suitable for practical implementations where storage and communication overhead must be minimized. Among the newer schemes, ZZCG25 introduced by Zhu et al. (2025) also offers compact parameters with ciphertext and

helper key sizes of $\mathcal{P}(\lambda, \log n)$, while its public parameters scale with circuit complexity ($\mathcal{P}(\lambda, k, l_c, \log n)$). This represents an improvement over schemes like AT24 introduced by Attrapadung and Tomida (2024), which requires significantly larger parameters scaling with $|\mathcal{U}|^2$ or $n|\mathcal{U}|$.

Regarding universe support, several recent schemes including FWW23 introduced by Freitag et al. (2023), AT24 introduced by Attrapadung and Tomida (2024), and ZZCG25 introduced by Zhu et al. (2025) support unbounded attribute universes, distinguishing them from earlier schemes like HLWW23 introduced by Hohenberger et al. (2023), ZZGQ23 introduced by Zhu et al. (2023), and GLWW24 introduced by R. Garg et al. (2024) that rely on bounded universes. The unbounded design in FWW23 comes at the cost of working in the ROM and relying on LWE assumptions, whereas ZZCG25 achieves unbounded universe support in the standard model. CHW25 introduced by Champion et al. (2025) takes a different approach by focusing on key-policy based RABE (K-General) but requires ROM security and relies on l-LWE assumptions.

Regarding computation costs, ZZZG24 introduced by Y. Zhang et al. (2024) offers superior efficiency, as Setup, KeyGen, Reg, Update, Enc, and Dec can all be executed in $\mathcal{P}(\log n)$ time, which is highly desirable for large systems. By contrast, earlier schemes like HLWW23 introduced by Hohenberger et al. (2023) and ZZGQ23 introduced by Zhu et al. (2023) require $\mathcal{P}(n)$ or $n \cdot \mathcal{P}(\lambda, \log n)$ computations in key generation, registration, and decryption, potentially limiting their scalability in large user bases or high-churn environments. Among the newer schemes, DPY24-ABE introduced by Datta et al. (2025) achieves relatively efficient encryption and decryption that scale with policy size ($\mathcal{P}(|P|)$), but its setup and registration costs remain dependent on attribute universe size. ZZCG25 introduced by Zhu et al. (2025) and CHW25 introduced by Champion et al. (2025) present different trade-offs: ZZCG25 has update and decryption costs that

scale linearly with n, while CHW25's operations are parameterized by lattice generation function $\mathcal{L}(n)$.

From a functionality and security perspective, the landscape has diversified considerably. While HLWW23 introduced by Hohenberger et al. (2023), ZZGQ23 introduced by Zhu et al. (2023), and GLWW24 introduced by R. Garg et al. (2024) support monotone span programs (Span) and arithmetic branching programs (ABP), newer schemes like AT24 introduced by Attrapadung and Tomida (2024) and ZZCG25 introduced by Zhu et al. (2025) support general policies, providing greater expressiveness in access control. DPY24-ABE introduced by Datta et al. (2025) uses span programs but operates in the generic ROM (G-ROM), representing a different security trade-off. Notably, ZZCG25 introduced by Zhu et al. (2025) achieves general circuit policies while maintaining standard model security under LWE assumptions and supporting unbounded universes, marking a significant advancement in RABE functionality.

From a methodological perspective, FWW23 introduced by Freitag et al. (2023) is the only scheme based on non-black-box use of cryptographic techniques, which introduces substantial computational overhead. In contrast, all other schemes in the table, including the newest entries AT24 introduced by Attrapadung and Tomida (2024), ZZCG25 introduced by Zhu et al. (2025), and CHW25 introduced by Champion et al. (2025), employ black-box techniques, enabling more efficient implementations.

The number of updates ($\#(Upd)$) remains consistently at $\mathcal{P}(\log n)$ across all schemes, which is raised by the standard construction approach, i.e. slotted approach. This contrasts with advancements seen in RBE schemes, where some constructions have achieved sub-logarithmic or even constant update complexity.

In conclusion, while ZZZG24 introduced by Y. Zhang et al. (2024) represents one of the most efficient and compact RABE solutions for bounded universes, newer schemes

like ZZCG25 introduced by Zhu et al. (2025) offer compelling alternatives by supporting unbounded universes and general policies while maintaining standard model security, albeit with some computation overhead in updates and decryption. AT24 introduced by Attrapadung and Tomida (2024) provides general policy support with unbounded universes but at the cost of large parameter sizes. CHW25 introduced by Champion et al. (2025) introduces a key-policy based approach that could be valuable for specific applications. DPY24-ABE introduced by Datta et al. (2025) offers a balance between parameter size and computation efficiency but requires G-ROM security. This diverse landscape of RABE schemes provides implementers with various options to choose from based on their specific requirements for parameter efficiency, computation cost, policy expressiveness, universe support, and security guarantees.

3.6 Registered Functional Encryption (RFE)

Registered Functional Encryption (RFE) like Datta and Pal (2023); Francati et al. (2023) is a robust cryptographic primitive, which includes RBE and RABE, extends the registration settings to Functional Encryption (FE). In an FE scheme, there is a PKG which holds a master secret key. The PKG generates secret keys sk_f for users. An encryption of a message m can be decrypted using sk_f to recover $f(m)$. FE overcomes the limitation of "all-or-nothing" decryption property of traditional public key encryptions (PKE) by delivering fine-grained access control and computing capability over encrypted data.

Similary to RBE and RABE, to address key-escrow issue, RFE replaces the PKG by an honest-but-curious KC. In RFE, a user registers a public key with a specific function f and the KC updates the public parameters and updates a helper key for the new user. This helper key allows the user's secret key to decrypt a ciphertext ct of x under the updated public parameters to $f(x)$. Existing users might also request to update helper

keys in the system. Same as RBE and RABE, FE encompasses four key entities: the Cloud Service Provider (CSP), the Key Curator (KC), the Data Owner (DO), and the Data User (DU).

3.6.1 Syntax. Definition (Syntax of RFE). Let $\mathcal{U}_F = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ be the universe of functions. A registered functional encryption (RFE) scheme consists of PPT algorithms (Gen, Reg, Enc, Upd, Dec) working as follows. The Reg and Upd algorithms are performed by the KC.

- Setup. $Setup(\lambda, l_f) \rightarrow crs$. The probabilistic algorithm takes the security parameter λ and the size l_f of the functions in $\mathcal{U}_F$ as inputs and outputs a common random string crs. Some of the subroutines below will need a common random string crs, which could be sampled publicly using some public randomness beacon.

- Key generation. $KGen(crs, aux) \rightarrow (pk, sk)$. The probabilistic algorithm KGen takes as input the common random string crs and a (possibly empty) auxiliary information aux and outputs a pair of public/secret keys (pk, sk). Note that these are only public and secret keys, not the encryption or decryption keys. The key generation algorithm is run by any honest user locally who wants to register itself into the system.

- Registration. $Reg(crs, aux, pk, f_{pk}) \rightarrow (pp, aux')$. The deterministic algorithm Reg takes as input the common random string crs, a (possibly empty) auxiliary information aux, a public key pk, and a function $f_{pk} \in \mathcal{F}_\lambda$, and it outputs the public parameters pp and an updated auxiliary information aux'.

- Encryption. $Enc(pp, m) \rightarrow ct$. The randomized algorithm Enc takes as input the public parameters pp and a plaintext message m and outputs a ciphertext ct.

– Update. $Upd(crs, aux, pk) \rightarrow hpk$. The deterministic algorithm Upd takes as input the common random string crs, an auxiliary information aux and a user public key pk, and generates a helper key hpk.

– Decryption. $Dec(sk, hpk, ct) \rightarrow m$: The deterministic decryption algorithm Dec takes as input a secret key sk, a helper key hpk, and a ciphertext ct, and it outputs a message m.

Definition (Compactness and Efficiency of RFE).

– **Compactness** The public parameters and helper keys must be short, i.e., $|pp| = \mathcal{P}(\lambda, l_f, \log i)$ for all $i \in [n]$ and $|hpk| = \mathcal{P}(\lambda, l_f, \log n)$, where λ is the security parameter, n is the number of registered users, and l_f is the size of functions in $\mathcal{U}_F$.

– **Efficiency of updates:** The number of times that a registered user must update helper keys must be low, i.e., $\mathcal{P}(\log n)$ over the lifetime of the system.

3.6.2 Adversarial Model and Security Goals. Follwing RBE, KCs in RFE are also considered semi-trusted and transparent. Besides, a secure RFE system must satisfy the following security properties:

– Data Confidentiality : in a RFE system, if a user is unauthorized user, it should be blocked from decrypting the ciphertext. In addition, the KC should not be allowed to decrypt ciphertexts without authorization. Furthermore, in a RFE system, an authorized user should only gain the function value $f(m)$ from the ciphertext and nothing else about the message m.

– Collusion Resistance: a RFE system should prevent collusion attacks from unauthorized users and the KC.

3.6.3 Research Status. Datta and Pal (2023) introduced the first notion of registered functional encryption (RFE) to address the key-escrow issue in functional encryption (FE) while ensuring both compactness and efficiency. They defined RFE to require that the sizes of the aggregated public key, helper decryption keys, ciphertexts, as well as the encryption and decryption times, remain polylogarithmic in the number of registered users. Additionally, their framework ensures that the KC is entirely transparent and does not maintain any secret information. They further proposed an RFE scheme supporting general functions and an arbitrary number of users, relying on indistinguishability obfuscation (IO) and somewhere statistically binding hash (SSBH) functions.

Francati et al. (2023) later presented a more efficient RFE construction, also based on IO and SSBH functions, but optimized for a general class of functions. Their scheme achieves significant improvements in compactness, as the common reference string (crs), public parameters, and helper keys are all of size $\mathcal{P}(\lambda)$. Moreover, the key generation process runs in $\mathcal{P}(\lambda)$ time, while registration operates in $n \cdot \mathcal{P}(\lambda)$, demonstrating enhanced scalability compared to previous approaches.

Branco et al. (2025) extended RFE to traitor-tracing systems demonstrated by Agrawal, Kumari, Yadav, and Yamada (2023); Chor, Fiat, and Naor (1994); Kim and Wu (2020) and constructed schemes based on bilinear maps. One of their schemes achieves adaptive security in the generic group model, producing ciphertexts of size $\mathcal{P}(\sqrt{n})$ group elements for quadratic functions. Meanwhile, their second scheme, which supports linear functions, achieves selective security under a static q-type assumption, providing additional flexibility in cryptographic constructions.

Zhu et al. (2024) developed the first RFE schemes specifically designed for linear function or inner-product like J. Chen, Gong, Kowalczyk, and Wee (2018); J. Chen,

Gong, and Wee (2018); Okamoto and Takashima (2016) evaluation using pairings. Their scheme achieves adaptive IND-security under the k-Lin assumption in prime-order bilinear groups. Additionally, they constructed the first RFE scheme supporting quadratic functions, which attains very selective simulation-based security (SIM-security) under the bilateral k-Lin assumption. Here, "very selective" security implies that the adversary must declare challenge messages, all quadratic functions to be registered, and all corrupted users at the outset of the attack.

Chu et al. (2024) further expanded the scope of RFE by proposing two pairing-based schemes for inner-product and quadratic functions, respectively. Their inner-product RFE scheme relies on the Matrix Decision Diffie-Hellman (MDDH) assumption Escala, Herold, Kiltz, Ràfols, and Villar (2017), while their quadratic RFE scheme is based on both the MDDH assumption and the bilateral MDDH (bi-MDDH) assumption. Both schemes achieve weakly weakly selective-IND security and weakly selective-SIM security, offering a trade-off between security and efficiency.

Following this, Datta et al. (2025) proposed the first RFE construction that supports non-predicate-based functionalities and linear functions. Building upon this foundation, they introduced the first Registered Attribute-Based Inner-Product Functional Encryption (RABIPFE) scheme, which enables access policies represented by Linear Secret Sharing Schemes (LSSS). Their construction is based on asymmetric prime-order bilinear groups and is proven secure in the generic bilinear group model. In addition, they presented a general-function RFE scheme using a black-box approach that combines indistinguishability obfuscation (iO) and SSB hash functions.

Later, Y. Zhang, Chen, He, and Zhang (2025) introduced a new unbounded collusion-resilient RFE for all polynomial-sized circuits, relying garbled circuits and global slotted registered broadcast encryption. Their construction is proved adaptively

secure in random oracle model, based on LWE and evasive LWE assumptions like Wee (2022). They achieve better compactness by relaxing lower collusion bound and allowing master public key and helper key of size being polynomial of the size of circuit, colusion bound, and $\log(n)$.

Overall, recent advancements in RFE have led to more compact, scalable, and expressive schemes, paving the way for practical applications in access control, privacy-preserving machine learning, and secure data sharing in large-scale systems.

3.6.4 Comparison.

Table 3. Comparisons of RFE schemes.

Schemes	**Parameter Size**					$\#(Upd)$
	$\lvert crs\rvert$	$\lvert aux\rvert$	$\lvert pp\rvert$	$\lvert ct\rvert$	$\lvert hpk\rvert$	
DP23 Datta and Pal (2023)	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	-	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\log n)$
FFMM23 Francati et al. (2023)	$\mathcal{P}(\lambda)$	-	$\mathcal{P}(\lambda)$	$(\lvert sk\rvert + \lvert \rfloor\sqcup\rvert)\cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$
BLMM24-Q Branco et al. (2025)	$\mathcal{P}(\lambda, n_2, \log L)$	-	$\mathcal{P}(\lambda, n_1 + n_2)$	$\mathcal{P}(\lambda, n_1 + n_2)$	$\mathcal{P}(\lambda, n_1 n_2)$	$\mathcal{P}(\log L)$
BLMM24-L Branco et al. (2025)	$\mathcal{P}(\lambda)$	-	$\mathcal{P}(\lambda, \sqrt{L})$	$\mathcal{P}(\lambda, \sqrt{L})$	1	$\mathcal{P}(\log L)$
ZLZG24-L Zhu et al. (2024)	$\mathcal{P}(\lambda, \lvert m\rvert^2, L^2)$	-	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\log L)$
ZLZG24-Q Zhu et al. (2024)	$\mathcal{P}(\lambda, n_1, n_2, \log L)$	-	$\mathcal{P}(\lambda, n_1 + n_2)$	$\mathcal{P}(\lambda, n_1, n_2)$	$\mathcal{P}(\lambda, \log L)$	$\mathcal{P}(\log L)$
CLQC24-I Chu et al. (2024)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	-	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\log n)$
CLQC24-Q Chu et al. (2024)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	-	$\mathcal{P}(\lambda, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\log n)$
DPY24-1 Datta et al. (2025)	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	-	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, \lvert m\rvert)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$

Continued on next page

Table 3. Comparisons of RFE schemes (Continued)

Schemes	**Parameter Size**					$\#(Upd)$
	$\lvert crs\rvert$	$\lvert aux\rvert$	$\lvert pp\rvert$	$\lvert ct\rvert$	$\lvert hpk\rvert$	
DPY24-2 Datta et al. (2025)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \log n)$	-	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, l_c)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\log n)$
ZCHZ25 Y. Zhang et al. (2025)	$\mathcal{P}(\lambda, l_c, Q, \log n)$	-	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\log n)$

Schemes	**Computation Cost**					
	Setup	KeyGen	Reg	Update	Enc	Dec
DP23 Datta and Pal (2023)	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	1	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\log n)$	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, \log \lvert\mathcal{U}_F\rvert, \log n)$
FFMM23 Francati et al. (2023)	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda)$	$n \cdot \mathcal{P}(\lambda)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda)$
BLMM24-Q Branco et al. (2025)	$\mathcal{P}(\lambda, n_2, \log L)$	$\mathcal{P}(\lambda, n_1, \log L)$	$\mathcal{P}(\lambda, n_1 + n_2)$	$\mathcal{P}(\lambda, n_1 n_2)$	$\mathcal{P}(\lambda, n_1 + n_2)$	$\mathcal{P}(\lambda, n_1 + n_2)$
BLMM24-L Branco et al. (2025)	1	$\mathcal{P}(\lambda, L)$	$\mathcal{P}(\lambda, \sqrt{L})$	1	$\mathcal{P}(\lambda, \sqrt{L})$	$\mathcal{P}(\lambda, \lvert x\rvert, \lvert m\rvert, \log L)$
ZLZG24-L Zhu et al. (2024)	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log L)$
ZLZG24-Q Zhu et al. (2024)	$\mathcal{P}(\lambda, n_1, n_2, \log L)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, n_1, n_2, \log L)$	$\mathcal{P}(\lambda, n_1, n_2)$	$\mathcal{P}(\lambda, \log L)$	$\mathcal{P}(\lambda, n_1, n_2)$
CLQC24-I Chu et al. (2024)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$
CLQC24-Q Chu et al. (2024)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert m\rvert, n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$
DPY24-1 Datta et al. (2025)	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert m\rvert)$	$\mathcal{P}(\lambda, \lvert m\rvert)$
DPY24-2 Datta et al. (2025)	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, \lvert\mathcal{U}_F\rvert, \log n)$	$\mathcal{P}(\lambda, l_c, \lvert m\rvert)$	$\mathcal{P}(\lambda, l_c, \lvert m\rvert)$
ZCHZ25 Y. Zhang et al. (2025)	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$	$\mathcal{P}(\lambda, l_c, Q, \log n)$

Schemes	**black-box**	**Universe**	**Group**	**Func**	**Security**	**Assump**

Continued on next page

Table 3. Comparisons of RFE schemes (Continued)

Schemes	Parameter Size					#(Upd)
	$\|crs\|$	$\|aux\|$	$\|pp\|$	$\|ct\|$	$\|hpk\|$	
DP23 Datta and Pal (2023)	•	UnB	-	General	STM	IO, SSBH
FFMM23 Francati et al. (2023)	○	UnB	Prime	General	STM	IO, SSBH
BLMM24-Q Branco et al. (2025)	•	UnB	Prime	Quadratic	G-ROM	-
BLMM24-L Branco et al. (2025)	•	B	Prime	Linear	ROM	q-type
ZLZG24-L Zhu et al. (2024)	•	B	Prime	Linear	STM	k-Lin
ZLZG24-Q Zhu et al. (2024)	•	B	Prime	Quadratic	STM	k-Lin
CLQC24-I Chu et al. (2024)	•	B	Prime	Inner Prod	STM	MDDH
CLQC24-Q Chu et al. (2024)	•	B	Prime	Quadratic	STM	MDDH, bi-MDDH
DPY24-1 Datta et al. (2025)	•	B	Prime	Linear	G-ROM	-
DPY24-2 Datta et al. (2025)	•	UnB	-	General	STM	IO, SSBH
ZCHZ25 Y. Zhang et al. (2025)	•	UnB	-	General	ROM	LWE

Notes:

n_1, n_2: dimensions of message space; L: number of slots; $|\mathcal{U}_F|$: function space size.

$|m|$: bit length of a message; n: number of users; l_c: length of a binary circuit.

Q: collusion bound; STM: standard security; ROM: random oracle model.

Table 3 provides a comprehensive comparison of various RFE schemes based on parameter sizes, computational costs, security properties, and underlying assumptions. This section identifies the most efficient scheme in each category.

From the parameter size perspective, FFMMR23 introduced by Francati et al. (2023) and BLMM24-L introduced by Branco et al. (2025) exhibit the most compact designs, with public parameters and ciphertexts bounded by $\mathcal{P}(\lambda)$ or $\mathcal{P}(\lambda, \sqrt{L})$, making them highly suitable for practical deployments in constrained environments. DPY24-

1 introduced by Datta et al. (2025) also shows promise with a concise $\mathcal{P}(\lambda)$ public parameter size, though its ciphertext size scales with message length as $\mathcal{P}(\lambda, |m|)$. In contrast, ZLZG24-L introduced by Zhu et al. (2024) and CLQC24 introduced by Chu et al. (2024) show larger parameter dependencies, involving quadratic terms like $|m|^2$ and L^2, which may limit scalability when either message length or the number of slots becomes large. Notably, DP23 introduced by Datta and Pal (2023) and ZCHZ25 introduced by Y. Zhang et al. (2025) achieve good balance by keeping parameters logarithmic in their respective function spaces ($|\mathcal{U}_F|$ or l_c), though ZCHZ25 introduces additional dependency on the collusion bound Q.

Regarding computation costs, BLMM24-L introduced by Branco et al. (2025) stands out for its highly efficient setup and update operations, both constant time (1), and for achieving square-root complexity in other algorithms (e.g., $\mathcal{P}(\lambda, \sqrt{L})$ in KeyGen, Reg, Enc), making it favorable for systems needing frequent dynamic updates. DPY24-1 introduced by Datta et al. (2025) maintains consistent $\mathcal{P}(\lambda, |m|, \log n)$ efficiency across most operations, with slightly improved encryption and decryption costs at $\mathcal{P}(\lambda, |m|)$, presenting a balanced choice for systems with moderate message sizes. FFMMR23 introduced by Francati et al. (2023), while having simple $\mathcal{P}(\lambda)$ complexity in many phases, suffers from $n \cdot \mathcal{P}(\lambda)$ cost in registration, which could become a bottleneck in large-scale deployments. By comparison, ZCHZ25 introduced by Y. Zhang et al. (2025) maintains uniform cost $\mathcal{P}(\lambda, l_c, Q, \log n)$ across all operations, making its performance more predictable but potentially heavier for complex circuits with large l_c.

For update efficiency ($\#(Upd)$), all schemes achieve logarithmic complexity—$\mathcal{P}(\log n)$ or $\mathcal{P}(\log L)$—which is essential for scalability in dynamic user environments. This consistency across schemes highlights the importance of efficient revocation and update mechanisms in practical RFE deployments.

Examining functionality and expressiveness, recent advances show interesting trends. DPY24-2 introduced by Datta et al. (2025) joins DP23 introduced by Datta and Pal (2023) and FFMMR23 introduced by Francati et al. (2023) in supporting general functional encryption over an unbounded universe, offering maximal flexibility but requiring strong cryptographic assumptions (IO, SSBH). ZCHZ25 introduced by Y. Zhang et al. (2025) represents a significant advancement by supporting general functionalities while relying on more standard lattice-based assumptions (LWE) rather than IO, though in the Random Oracle Model (ROM). This marks an important step toward practical general-purpose RFE. Meanwhile, DPY24-1 introduced by Datta et al. (2025) provides an efficient solution for linear functions with G-ROM security, complementing BLMM24-L's approach with different efficiency tradeoffs.

More practical constructions such as BLMM24 and ZLZG24 focus on specific function classes like quadratic and linear functions, striking a balance between expressiveness and efficiency. Notably, CLQC24 introduced by Chu et al. (2024) is unique in supporting inner product (IP) and quadratic functions with STM security, under more standard assumptions (MDDH, bi-MDDH), making it a strong candidate for applications needing strong security guarantees in the standard model.

From a security standpoint, the landscape shows diversity in approaches. ZCHZ25 introduced by Y. Zhang et al. (2025) introduces a novel post-quantum secure option based on LWE, though in the ROM, addressing the need for quantum-resistant RFE solutions. DPY24 offers two constructions with different security-efficiency tradeoffs: DPY24-1 achieves efficiency in the G-ROM without specific assumptions, while DPY24-2 provides STM security with stronger assumptions (IO, SSBH). Earlier schemes like ZLZG24 introduced by Zhu et al. (2024) and CLQC24 introduced by Chu et al. (2024)

maintain their relevance for operating in the standard model with STM security under well-established assumptions (e.g., k-Lin, MDDH).

In conclusion, the choice of an RFE scheme depends on application requirements. For general functionality and unbounded universes with post-quantum considerations, ZCHZ25 offers a compelling option based on LWE, though in the ROM. If general functionality is needed with classical security, DP23, FFMMR23, and DPY24-2 are suitable options, with tradeoffs in assumption strength (IO, SSBH). For efficiency and specific function classes (linear, quadratic), BLMM24 and ZLZG24 provide practical solutions, with BLMM24-L and DPY24-1 achieving remarkable efficiency for linear functions, and BLMM24-Q and ZLZG24-Q supporting quadratic functions efficiently. CLQC24, supporting inner product and quadratic functionalities under standard assumptions, balances functionality, efficiency, and security.

Future research directions should focus on reducing parameter and computation costs for general functionalities, achieving standard model security for unbounded function spaces, and further exploring post-quantum RFE constructions like ZCHZ25. The evolving landscape suggests promising paths toward more practical and secure RFE systems suitable for diverse application scenarios.

3.7 Unified Analysis and Design Insights

The evolution of Registered Encryption (RE) reveals core design patterns, persistent bottlenecks, and critical strategic trade-offs. These insights transcend individual constructions, synthesizing cross-cutting trends that illuminate RE's maturation as a cryptographic primitive.

3.7.1 Architectural Evolution: From Non-Black-Box to Black-Box Paradigms.

A significant trend in RE development has been the shift from non-black-box constructions to black-box designs. Early RE constructions, like those by S. Garg

et al. (2018), required deep inspection of cryptographic components. They relied on non-black-box techniques, such as indistinguishability obfuscation (IO) and complex bilinear pairing manipulations. This necessitated intricate analysis of internal encryption algorithms, making security proofs complex and implementations challenging.

Modern RE schemes increasingly treat underlying primitives as modular components. They define well-defined security interfaces for these components. This shift offers several benefits:

- Enhanced Modularity: It allows for a cleaner separation of concerns.
- Post-Quantum Readiness: It naturally accommodates diverse cryptographic assumptions.
- Reduced Implementation Complexity: It avoids deep algebraic manipulations.

However, this shift often introduces a fundamental trade-off. Black-box constructions typically have larger parameter sizes and reduced efficiency, balancing theoretical elegance against practical performance.

3.7.2 The Helper Key Update Bottleneck. Helper key update mechanisms consistently emerge as the primary scalability bottleneck across nearly all RE constructions. This issue appears inherent to the registration paradigm itself. The bottleneck stems from the tension between update frequency and security. Most schemes require helper key updates with each registration to maintain security against adaptive attackers.

Frequent updates ensure strong security but impose substantial overhead on all participants. Conversely, batched updates improve efficiency but create vulnerability windows. Furthermore, helper key updates necessitate broadcasting parameters to all participants, leading to significant broadcast overhead. In large-scale deployments, this

becomes prohibitive with high registration rates, as users must maintain fresh keys to decrypt new messages.

Maintaining consistent helper key states across distributed systems also creates significant synchronization challenges. Users missing updates lose decryption capabilities, and systems must handle participants operating with incompatible key versions. Recent mitigation strategies, including tree-based structures, lazy updates, and approximate synchronization, attempt to address these issues. However, they invariably involve trade-offs between security, efficiency, and overall complexity.

3.7.3 The Public Parameter Distribution Challenge. Public parameter management represents another fundamental scalability bottleneck. Unlike helper keys, public parameters must be universally shared among all entities, yet they require consistent updates following registrations. These universal distribution requirements mean that public parameters encode the global system state, including aggregate user information and cryptographic elements. Thus, every registration potentially affects every participant, creating global dependencies that fundamentally limit scalability.

A conflict arises between consistency and availability. Universal parameter consistency often clashes with system availability, as synchronous updates can render systems unavailable. Asynchronous propagation, meanwhile, creates consistency windows that may cause delivery failures. This dynamic also leads to network amplification effects. Each registration triggers updates propagating to all participants, leading to quadratic bandwidth growth with system scale. This poses a fundamental scalability ceiling for RE deployments.

Finally, state synchronization challenges emerge from out-of-order updates or partial applications. Such issues can create inconsistent states where operations fail unpredictably, requiring sophisticated consensus mechanisms that add substantial

implementation complexity. Recent work has explored hierarchical parameter structures, lazy propagation mechanisms, and approximate consensus protocols. Still, these solutions invariably involve fundamental trade-offs between consistency guarantees, system performance, and implementation complexity.

3.7.4 Cryptographic Assumptions. The choice of underlying cryptographic assumptions profoundly impacts RE design. Learning With Errors (LWE) and Indistinguishability Obfuscation (IO) represent fundamentally different design philosophies.

LWE-based constructions, built on lattice problems, offer several advantages:

- Post-quantum security.
- Relatively efficient operations.
- Well-understood security foundations.

However, they typically suffer from large parameter sizes, particularly for Common Reference String (CRS) and helper key representations.

In contrast, IO-based constructions, relying on indistinguishability obfuscation, offer greater design flexibility and often more compact representations. Yet, they come with enormous concrete efficiency costs and rely on less-understood assumptions. Increasingly, modern RE schemes adopt hybrid approaches, combining benefits from different assumption classes. For instance, they might use IO for complex one-time setups while relying on more standard assumptions for routine operations. The assumption choice also directly influences post-quantum readiness; while LWE-based constructions provide clear post-quantum security, the post-quantum status of IO-based schemes remains uncertain.

3.7.5 Security-Efficiency Trade-off Patterns.

Analysis across RE constructions consistently reveals specific security-efficiency trade-off patterns fundamental to the registration paradigm.

This includes the relationship between CRS size and update frequency. Schemes with a compact CRS typically require more frequent updates to maintain security. Conversely, constructions allowing infrequent updates often demand significantly larger CRSs. This directly impacts deployment scenarios based on storage and bandwidth constraints.

Another observed pattern is the inverse relationship between registration and encryption efficiency. Constructions optimized for fast registration often impose additional computational costs on encryption operations. Schemes with efficient encryption typically necessitate more complex registration procedures.

Finally, the security model strength versus performance demonstrates a clear trend: stronger security models, particularly adaptive security, consistently come at substantial efficiency costs. The performance gap between selective and adaptive security often spans orders of magnitude.

3.7.6 Design Principles.

This unified analysis reveals emerging design principles fundamental to successful RE construction:

- **Hierarchical Update Structures:** Successful scalable schemes increasingly adopt hierarchical approaches to helper key updates. They use tree-based structures to reduce broadcast overhead while maintaining security guarantees.

- **Assumption Diversification:** Mature designs incorporate multiple assumption classes to balance security, efficiency, and post-quantum readiness.

- **Adaptive Parameter Selection:** Advanced schemes provide mechanisms for dynamic parameter adjustment based on system scale, usage patterns, and security requirements.

- **Graceful Degradation:** Robust designs incorporate mechanisms for maintaining core functionality even when update synchronization fails or system components become unavailable.

These insights mark RE's emergence as a mature cryptographic primitive with well-understood capabilities and limitations.

3.8 Future Research Directions

While RE has achieved significant theoretical and practical milestones, several fundamental challenges and promising extensions remain unexplored or only partially addressed. This section outlines the most critical research directions that will shape the next generation of RE systems, from strengthening security guarantees to enabling new application domains.

3.8.1 Efficiency and Scalability Research.

The practical viability of RE systems fundamentally depends on achieving efficiency levels comparable to traditional encryption schemes while maintaining the unique security properties of the registration paradigm. This represents a critical research frontier with multiple interconnected challenges.

Registration Process Acceleration: The computational bottleneck in current RE systems often occurs during user registration, where new helper keys must be computed and distributed. Research into incremental update algorithms, parallel processing techniques for helper key generation, and amortized registration schemes that batch multiple registrations could significantly improve registration throughput. Furthermore,

developing predictive registration mechanisms that pre-compute potential helper key updates could reduce registration latency.

Ciphertext Size Minimization: RE ciphertexts are typically larger than traditional encryption schemes due to additional metadata required for the registration-based decryption process. Research into ciphertext compression techniques, optimized encoding schemes for those metadata, and hybrid approaches that selectively apply compression based on content characteristics could reduce storage and transmission costs.

Computational Complexity Reduction: Both encryption and decryption operations in current RE schemes involve significant computational overhead compared to traditional encryption. Research into algorithmic optimizations, specialized mathematical techniques for the registration setting, and hardware-accelerated implementations could significantly improve throughput. This includes developing RE-specific cryptographic primitives optimized for the registration paradigm rather than adapting general-purpose constructions.

Communication Efficiency: The distribution of helper keys and registration updates represents a significant communication overhead in RE systems. Research into efficient broadcast protocols, delta-based update mechanisms that only transmit changes rather than complete helper keys, and peer-to-peer distribution strategies could substantially reduce communication costs. Additionally, developing asynchronous update protocols that don't require simultaneous availability of all participants could improve practical deployability.

Scalability Analysis and Optimization: Comprehensive scalability research must address how RE performance degrades with increasing numbers of users, registration frequency, and potential functional request frequency (e.g., user revocation).

This includes developing theoretical models for RE scalability and optimization strategies specifically designed for large-scale deployment scenarios.

3.8.2 Verifiability and Accountability Mechanisms. Current RE schemes operate under the assumption that key curators behave honestly, but practical deployment requires to detect and respond to curator misbehavior. Developing comprehensive verifiability and accountability frameworks represents a critical research frontier.

Registration Verification: Users may require to verify that their registration has been correctly processed and that the resulting helper keys accurately reflect the current registration state. This requires developing efficient zero-knowledge proofs or similar mechanisms that allow curators to demonstrate correct helper key computation without revealing sensitive information about other registrations.

Audit Mechanisms: Long-term RE deployment requires comprehensive audit capabilities that allow retrospective verification of curator behavior. This includes mechanisms for detecting unauthorized access attempts, incorrect helper key updates, or other forms of curator misbehavior. Developing efficient audit trails that preserve privacy while enabling accountability represents a significant technical challenge.

Distributed Curator Models: Single curator systems present natural targets for attack and coercion. Research into distributed curator architectures, where multiple parties collectively maintain the registration system without any single party having complete control, could significantly enhance both security and accountability. This requires sophisticated threshold cryptography and consensus mechanisms adapted to the registration paradigm.

Cryptographic Accountability: Beyond detecting misbehavior, RE systems should provide cryptographic evidence of curator misconduct that can be verified by third

parties. This involves developing non-repudiation mechanisms and cryptographic proofs that withstand legal scrutiny while ensuring system efficiency and privacy guarantees.

3.8.3 Advanced Key Management: Revocation, Delegation, and Temporal Controls.

Current RE schemes provide limited support for advanced key management operations that are essential for practical deployment. Developing efficient mechanisms for key revocation, delegation, and time-based access control represents a crucial research direction.

Efficient Revocation Mechanisms: User key revocation like the mechanism introduced by Ge et al. (2021) in RE systems currently requires expensive helper key updates that affect all system participants. Research into more efficient revocation mechanisms, such as revocation trees, accumulator-based approaches, or lazy revocation schemes, could dramatically improve scalability. The challenge lies in maintaining the escrow-free property while enabling efficient revocation without requiring universal helper key updates.

Delegation Frameworks: Enabling users to delegate decryption capabilities to other parties while maintaining fine-grained control over the scope and duration of delegation would significantly expand RE applicability. This requires developing delegation mechanisms like Mu, Varadharajan, and Quac Nguyen (1999); Porwal and Mittal (2023) that preserve the registration-based security model while preventing unauthorized delegation chains or capability escalation.

Time-Based Encryption: Incorporating temporal access controls like F. Zhao, Weng, Xie, Hou, and Li (2024), where ciphertexts can only be decrypted during specific time periods or after certain deadlines, adds significant practical value. Time-based RE requires careful coordination between temporal parameters and helper key update schedules, along with mechanisms to prevent temporal manipulation attacks.

Hierarchical Access Control: Extending RE to support hierarchical access structures like Horwitz and Lynn (2002); G. Wang, Liu, and Wu (2010), where users with higher privileges can access content intended for lower privilege levels, requires sophisticated key derivation mechanisms that maintain the escrow-free property across privilege levels.

3.8.4 Extensions to Advanced Cryptographic Primitives.

The success of the registration paradigm in access control encryption schemes naturally leads to explorations of its applicability to more sophisticated cryptographic primitives. Each such extension introduces distinct research challenges and opportunities, promising to significantly broaden the scope of secure data management and computation.

Registered Searchable Encryption (RSE): Extending the RE paradigm to support searchable encryption necessitates novel approaches for index generation and query processing that uphold both search privacy and the escrow-free property inherent in registration demonstrated like Boneh, Di Crescenzo, Ostrovsky, and Persiano (2004b); L. Meng, Chen, Tian, Manulis, and Liu (2024a). The primary challenge lies in constructing searchable encryption schemes where search capabilities are granted based on registered identities, critically without compromising data privacy or relying on centralized, trusted search index generators. This would enable secure and privacy-preserving querying of encrypted databases where access control is dynamically managed through registration.

Registered Broadcast Encryption (RBE): The integration of the registration paradigm with broadcast encryption schemes presents the challenge of efficiently managing helper keys across dynamic recipient sets like Fiat and Naor (1994); Freitag et al. (2023). For scenarios where content is encrypted for multiple recipients simultaneously, sophisticated techniques are required to handle recipient additions and

removals without necessitating complete helper key regeneration. This extension would significantly enhance the efficiency and scalability of secure content distribution in registered environments.

Registered Homomorphic Encryption (RHE): Perhaps the most ambitious extension involves synergizing registration-based access control with homomorphic computation capabilities like Acar, Aksu, Uluagac, and Conti (2018); Rivest, Adleman, Dertouzos, et al. (1978). Such a primitive would enable privacy-preserving computations directly on RE-encrypted data, while simultaneously maintaining fine-grained access control dictated by registered identities. The technical complexities at the intersection of these two advanced cryptographic primitives are substantial; however, the potential for transformative applications, particularly in secure cloud computing and data analytics, is immense.

Registered Multi-Party Computation (RMPC): Extending the registration paradigm to multi-party computation (MPC) protocols offers a pathway to privacy-preserving collaborative computation where participant involvement is explicitly controlled via registration mechanisms like Feng and Yang (2022). This requires the development of novel security frameworks for MPC protocols that can robustly accommodate dynamic participant sets and integrate registration-based access control seamlessly, opening new avenues for secure, collaborative data analysis among authorized parties.

3.8.5 Post-Quantum Registered Encryption.

The shift to post-quantum cryptography offers both challenges and opportunities for Registered Encryption (RE) research. While some RE constructions already use lattice-based assumptions for post-quantum security like Döttling et al. (2023), a full transition demands several key considerations. A crucial step is to diversify post-quantum cryptographic foundations

beyond current lattice-based methods. Developing RE schemes based on alternatives like code-based, hash-based, or multivariate cryptography is essential. This diversification provides vital security redundancies, protecting against potential cryptanalytic breakthroughs that could target specific foundational assumptions.

The move to a post-quantum environment will also require hybrid classical-quantum security schemes. These schemes must secure against both classical and quantum attacks while staying efficient for classical operations, demanding sophisticated ways to integrate classical and post-quantum cryptographic assumptions within the registration framework. Additionally, quantum-resistant parameter evolution is critical, as post-quantum cryptography often requires larger parameters. Research into these strategies is vital to maintain security while minimizing the impact on helper key sizes and update frequencies. Finally, key curator operations—both computational processes and communication protocols for helper key distribution—must be secure against quantum threats. Developing new approaches to authenticated communication and parameter distribution is paramount to ensure their continued security in a post-quantum setting.

3.8.6 Practical Deployment and Standardization. Beyond theoretical advances, the maturation of RE as a practical cryptographic primitive requires focused research on deployment considerations and standardization efforts.

Implementation Security: Developing secure implementations of RE schemes requires addressing side-channel attacks, fault injection vulnerabilities, and other implementation-specific security concerns that may not be captured in theoretical security models.

Interoperability Standards: Large-scale RE deployment will require standardized protocols for registration, helper key distribution, and cross-system

communication. Research into developing these standards while preserving the flexibility needed for ongoing cryptographic innovation represents a crucial practicality consideration.

Performance Optimization: Continued research into algorithmic and implementation optimizations could significantly enhance the practical performance of RE. This includes specialized approaches for specific deployment scenarios, such as mobile devices, IoT systems, or high-throughput cloud environments.

These research directions collectively represent the path toward mature, practical RE systems that can serve as foundational infrastructure for privacy-preserving applications. Success in these areas would establish RE not merely as an academic curiosity but as a transformative cryptographic primitive capable of reshaping how we approach identity-based security in both classical and post-quantum settings.

3.9 Conclusion

The escalating demand for privacy-preserving data sharing has underscored the vulnerabilities of traditional access-controlled encryption schemes, which depend on fully trusted authorities. RE offers a compelling alternative, decentralizing trust via a semi-trusted KC. By enabling users to independently generate key pairs and register identities, attributes, or functions without disclosing secrets, RE effectively resolves the pervasive key escrow problem inherent in IBE, ABE, and FE.

Our survey delivered the inaugural comprehensive overview of the RE landscape. We established a unified taxonomy classifying RE schemes. Tracing the historical trajectory from IBE to RE, we elucidated the paradigm's emergence and contrasted it with related escrow-free primitives. A rigorous assessment framework then evaluated the security, performance, and scalability of various RE designs. Through systematic analysis of state-of-the-art RBE, RABE, and RFE, we highlighted critical trade-offs across

ciphertext size, helper key updates, adversarial models, and complexity assumptions, providing invaluable insights into RE's current scope and limitations. Furthermore, we offered a unified analysis and design insights for RBE, encouraging researchers to apply the RE paradigm to other encryption schemes. Finally, we identified and mapped out critical future RE research directions, encompassing efficiency, scalability, functionality, integration with other cryptographic primitives, and post-quantum implementation.

As privacy regulations tighten and decentralized infrastructures—such as federated learning, edge computing, and blockchain systems—continue to proliferate, the need for cryptographic mechanisms that reduce trust dependencies will grow sharply. Registered Encryption is uniquely positioned to become a foundational primitive in this emerging ecosystem. By reconciling accountability with user autonomy and eliminating centralized key escrow, RE aligns with shifting trust models toward partial or zero trust and enables seamless interoperability with identity and attribute management systems. In doing so, it is likely to underpin the secure, flexible, and scalable architectures of future privacy-critical digital services.

This survey not only establishes the groundwork for sustained exploration of RE but also serves as a vital roadmap for researchers dedicated to pioneering the next generation of decentralized and trust-minimized cryptographic protocols.

Having established the theoretical landscape of registration-based encryption and identified the key escrow problem as a central obstacle, we now turn to the practical challenges of symmetric searchable encryption. In the next chapter, we address the most immediate threat in SSE deployments—volume leakage—by proposing the VLR-EDESE scheme, which fortifies symmetric constructions while maintaining the "easily deployable" characteristics essential for real-world cloud systems.

CHAPTER IV

SSE IN THE CENTRALIZED TRUST SETTING

This chapter is based on the paper *"Leakage-Resilient Easily Deployable and Efficiently Searchable Encryption (EDESE),"* published in the *Proceedings of the 30th ACM Symposium on Access Control Models and Technologies (SACMAT 2025)*. I was the primary author, responsible for the scheme design, security analysis, implementation, and manuscript preparation. Dr. Yingjiu Li provided conceptual guidance and editorial revisions. Jun Li, Daoyuan Wu, Jianting Ning, Yangguang Tian, and Robert H. Deng contributed to the refinement of the manuscript.

Building upon the comprehensive landscape of searchable encryption analyzed in Chapter II, we first address the most immediate challenge in symmetric settings: data leakage. While symmetric searchable encryption (SSE) offers the performance necessary for large-scale cloud systems, its vulnerability to leakage-abuse attacks (LAA) limits its adoption in sensitive sectors. In this chapter, we propose VLR-EDESE to fortify the security of symmetric constructions by mitigating volume leakage.

4.1 Introduction

Easily Deployable and Efficiently Searchable Encryption. The Easily Deployable and Efficiently Searchable Encryption (EDESE) is a promising cryptographic primitive designed specifically for practical symmetric searchable encryption (SSE) applications, e.g., Bost and Fouque (2017); W. He, Akhawe, Jain, Shi, and Song (2014); Lau et al. (2014); Song et al. (2000); Yuan, Li, Ning, and Deng (2022). It offers two advantages: *easy deployment* and *efficient search*. Compared to existing SSEs, easy deployment makes EDESE the optimal choice for protecting existing searchable services.

The easy deployment of EDESE stems from minimal server involvement: the server only stores encrypted data and performs simple matching. When a client sends

a query token for a keyword, the server matches it directly against stored ciphertexts using bitwise comparison—mirroring the process on unencrypted data. This compatibility enables seamless migration from existing systems to EDESE-based searchable services. It leads to significantly reduced deployment and maintenance costs, attracting both individual and enterprise customers.

Another key benefit of EDESE is its remarkable search efficiency on the server side, which leverages the inherent efficiency of SSE. Due to the streamlined matching process employed by EDESE, the search performance on the server side closely resembles that of searching over plaintext. In contrast, most existing searchable encryption schemes require the server to run complex test algorithms, which may involve decryption or other intricate mathematical operations For example, PIR-protected schemes like Angel, Chen, Laine, and Setty (2018); Hoang, Yavuz, Durak, and Guajardo (2017) require complex computations on the server side. Consequently, EDESE outperforms other searchable encryption schemes as the most efficient searchable encryption solution in practical scenarios.

Leakage-Abuse Attack. EDESE is vulnerable to Leakage-Abuse attacks (LAAs), widely studied in numerous research works, e.g., Blackstone, Kamara, and Moataz (2019); Cash, Grubbs, Perry, and Ristenpart (2015); Gui, Paterson, and Patranabis (2023); Islam, Kuzu, and Kantarcioglu (2012); Lambregts, Chen, Ning, and Liang (2022); C. Liu, Zhu, Wang, and Tan (2014); Ning et al. (2021); Oya and Kerschbaum (2022); Pouliot and Wright (2016). These attacks leverage leaked information from searchable encryption schemes, including EDESE, to infer sensitive data with the help of auxiliary information. In the context of searchable encryption literature, “leakage” refers to any information that an attacker can discern about query tokens, encrypted documents, and query responses. The

auxiliary information consists of a partial or complete set of source documents or a set of documents with a distribution similar to that of the source documents.

LAAs encompass various types of leakage profiles, such as access patterns, search patterns, volume patterns, co-occurrence patterns, etc. Specifically, the access pattern leakage exposes which encrypted document is associated with a query token. Search pattern leakage implies whether two query tokens correspond to the same keyword. Volume pattern leakage refers to the number of encrypted documents associated with a query token (or vice versa, the number of query tokens associated with an encrypted document). Co-occurrence pattern leakage illustrates how many encrypted documents are associated with a pair of query tokens or how many query tokens are associated with a pair of encrypted documents.

However, it is important to note that EDESE is more vulnerable to LAAs than general searchable encryption (SE) schemes due to its more stringent requirements. Its matching process on the server restricts the data owners from employing complex approaches to mitigating LAAs, and the only option for addressing LAAs is to incorporate dummy messages into the server's responses as a means of addressing such attacks.

Existing Approaches. Existing approaches to safeguard general SE from LAAs cannot protect EDESE because these approaches require additional server operations except for matching operations or multiple communications between server and client, and cannot be applied to EDESE, for example, Angel et al. (2018); Bost and Fouque (2017); Kamara and Papamanthou (2013); C. Liu et al. (2014); Patel, Persiano, Yeo, and Yung (2019); Song et al. (2000). On the other hand, existing countermeasures that fulfill the requirement of EDESE only focus on query volume leakage like Blackstone et al. (2019); Cash et al. (2015, 2013); Gui et al. (2023); Islam et al. (2012). They are not secure against

other leakages, such as document volume leakage demonstrated like Ning et al. (2021) and query/document co-occurrence leakage like Cash et al. (2015); Gui et al. (2023); Pouliot and Wright (2016).

4.1.1 Our Contribution. Leakage-Resilient EDESE.

To effectively strengthen EDESE against LAAs while maintaining a balance between efficiency and security, we introduce a novel security concept: **Leakage-Resilient EDESE (LR-EDESE)**, characterized by k-indistinguishability.

To avoid confusion, we clarify that "leakage-resilient" in this paper differs from its conventional cryptographic meaning, which typically refers to resistance against side-channel attacks that leak memory or computation details. Here, it signifies that an EDESE scheme remains secure against LAAs even when partial information is exposed. Additionally, our notion of "k-indistinguishability" aligns more closely with k-anonymity, ensuring that each query or document volume is indistinguishable from at least k alternatives, thereby limiting leakage exploitation. We rigorously formalize LR-EDESE within a cryptographic framework under this definition.

Volume Leakage Hiding Methods. We propose a novel volume leakage hiding method called modulo-based document partition (MDBPar), which partitions document volumes based on a modulo operation to hide document volume leakage, and a query volume leakage hiding method, inspired by Bost and Fouque (2017)'s work , named keyword clustering (KClu), which groups keywords into clusters to obscure query volume patterns. Both approaches introduce *dummy documents* alongside real keywords to effectively mask volume leakage. We formally prove that MDBPar and KClu achieve k-indistinguishability, ensuring robust mitigation of both document and query volumes in EDESE.

Volume Leakage-Resilient EDESE. We construct a concrete Volume Leakage-Resilient EDESE (VLR-EDESE). The VLR-EDESE achieves the complete query response, concurrently thwarting LAAs in terms of query volume and document volume leakages through the incorporation of k-indistinguishability. The VLR-EDESE sequentially employs the proposed volume leakage hiding methods to generate encrypted documents. Initially, VLR-EDESE applies the MDBPar method to derive a set of semi-obfuscated documents from the source documents. Then it takes this set as input for the subsequent execution of the KClu method, producing a set of fully obfuscated documents. These obfuscated documents comprise both original and dummy documents, and the server storage overhead of VLR-EDESE is linear to the count of the obfuscated documents.

Evaluations. We implement a prototype of VLR-EDESE and evaluate its performance through both theoretical analysis and experimental evaluation focusing on setup computation overhead, server storage overhead, and communication overhead. Our theoretical analysis demonstrates that VLR-EDESE achieves a setup complexity of $\mathcal{O}(m \log m + mn + km + k^2)$, a storage overhead of $\mathcal{O}(km + n)$, and a communication overhead of $\mathcal{O}(km + n)$, where m indicates the number of keywords, n denotes the number of documents, and k is the indistinguishability parameter. We conducted experiments using our prototype implementation on the Enron email corpus introduced by Shetty and Adibi (2004), a widely used benchmark in SSE research. The results highlight the efficiency of VLR-EDESE compared to state-of-the-art approaches, including PRT-EMM like Kamara and Moataz (2019) and FP-EMM like Patel et al. (2019). VLR-EDESE takes around 1 hour to produce encrypted documents based on the top 5,000 keywords and 37,172 source documents. With the strongest security setting ($k = 5000$), VLR-EDESE incurs an overall overhead of only 63× compared to the baseline EDESE without volume leakage mitigation, significantly lower than the 320× and 97× overheads

of adapted state-of-the-art schemes. This maximum k value represents the theoretical security limit achievable by any EDESE or ORAM-based scheme. Furthermore, we test with smaller k values (specifically 10, 20, 50, and 100), significantly reducing storage and communication overhead to no more than 2 times and 2.5 times that of the baseline scheme respectively. This demonstrates VLR-EDESE's flexibility in balancing efficiency and security.

CloudSec. We present CloudSec, which leverages the prototype implementation of VLR-EDESE to offer a user-friendly application tailored for the Windows platform. CloudSec enables seamless integration with various cloud storage platforms that provide REST APIs, such as OneDrive, Dropbox, Google Drive, and IDrive. In this paper, we highlight CloudSec's integration with Microsoft OneDrive, demonstrating its ability to interact effectively with cloud services.

Chapter Organization. The remainder of this chapter is organized as follows: Section 4.2 provides a comprehensive review of the literature on searchable encryption, highlighting the evolution of EDESE and its vulnerabilities to LAAs. Section 4.3 presents key concepts and definitions used in this work. Section 4.4 details the construction of our proposed VLR-EDESE scheme, while Section 4.5 provides formal security proofs. Section 4.6 evaluates the performance of VLR-EDESE through theoretical analysis and experimental results. Section 4.7 introduces CloudSec, our practical application of VLR-EDESE for cloud storage integration. Finally, Section 4.8 concludes the chapter and outlines future research directions.

4.2 Related Work

In this section, we conducted a comprehensive review of existing symmetric searchable encryption (SSE) schemes against LAAs. Our examination reveals two limitations when these schemes are applied to protect EDESE against LAAs:

Limitations in Applying SSE Schemes to EDESE. Firstly, existing SSE schemes, like Angel et al. (2018); Bost and Fouque (2017); Cash et al. (2013); Chase and Kamara (2010); G. Chen, Lai, Reiter, and Zhang (2018); Curtmola et al. (2006); Kamara and Moataz (2019); Kamara and Papamanthou (2013); C. Liu et al. (2014); Patel et al. (2019); Song et al. (2000), are often inadequate for protecting EDESE due to the nature of server operations extending beyond simple matching. SSE schemes designed to mitigate LAAs typically require additional server-side operations. For example, ORAM-protected SSE schemes like S. Garg, Mohassel, and Papamanthou (2016); Hoang et al. (2017); Kamara and Moataz (2019); Wu and Li (2023) introduce significant overhead through supplementary computations, state maintenance on the server, and increased client-server communication. However, EDESE primarily relies on the fundamental operations of a storage server, including storage and matching. In such cases, ORAM-protected SSE schemes fail to effectively safeguard EDESE. Designing appropriate EDESE schemes to defend LAAs is still challenging.

Limited Protection Coverage of Document Retrieval Phase. Many existing SSE schemes, e.g., Bost and Fouque (2017); Cash et al. (2013); G. Chen et al. (2018); Faber et al. (2015); Kamara and Moataz (2019); Kamara, Papamanthou, and Roeder (2012); Naveed, Prabhakaran, and Gunter (2014); Patel et al. (2019); Stefanov, Papamanthou, and Shi (2013) are categorized as structured encryption (STE) schemes, where relationships between keywords and documents are represented as mappings to document indices. While STE secures the index retrieval phase—allowing users to retrieve document indices matching a searched keyword—it introduces a potential vulnerability during document retrieval, when users access documents based on their indices. STE schemes, such as encrypted multi-maps (EMMs) like Chase and Kamara (2010); George et al. (2021); Kamara and Moataz (2019), enable outsourcing encrypted structural data to an untrusted

server while allowing the data owner to perform operations on the encrypted data. In these schemes, the ¡keyword, document credential¿ relationships, known as search indexes, are outsourced, and the client queries encrypted keywords (trapdoors) to retrieve document indices. This process, known as index retrieval, is crucial for SSE systems.

However, the query phase in EDESE includes both index and document retrieval, requiring protection for each to prevent leakage. Thus, using STE alone is insufficient. If EDESE protects only the search index with STE but naively outsources encrypted documents, attackers can observe retrieved documents and infer index patterns. To prevent this, our EDESE schemes securely outsource encrypted documents to mitigate leakage during document retrieval.

4.3 Definitions and Preliminaries

4.3.1 Pre-defined Notions. To illustrate proposed approaches more clearly, there are three predefined notions.

- Given a set of documents D, $K(D)$ denotes the set of all keywords in D. Besides, if given a set of encrypted documents ED, $K(ED)$ denotes all query tokens associated with ED.

- $Vec(D, d)$ represents all keywords associated with document $d \in D$. $Vec(D, kw)$ is defined to represent all documents associated with keyword $kw \in K(D)$. Similarly, $Vec(ED, ed)$ indicates all query tokens associated with an encrypted document $ed \in ED$ and $Vec(ED, qt)$ indicates all encrypted documents associated with a query token qt. In one word, Vec represents four functions based on its inputs.

- A volume function Vol is defined to denote the number of keywords (resp. query tokens) associated with a document (resp. encrypted document) or the number of

documents (resp. encrypted documents) associated with a keyword (resp. query tokens). This is referred to as query volume or document volume.

4.3.2 System Model of EDESE. EDESE involves two entities, a client and a server, and employs five core algorithms: key generation, setup, query token generation, matching, and final result algorithms. These algorithms are grouped into two distinct phases: setup phase and query phase, as shown in Fig. 1.

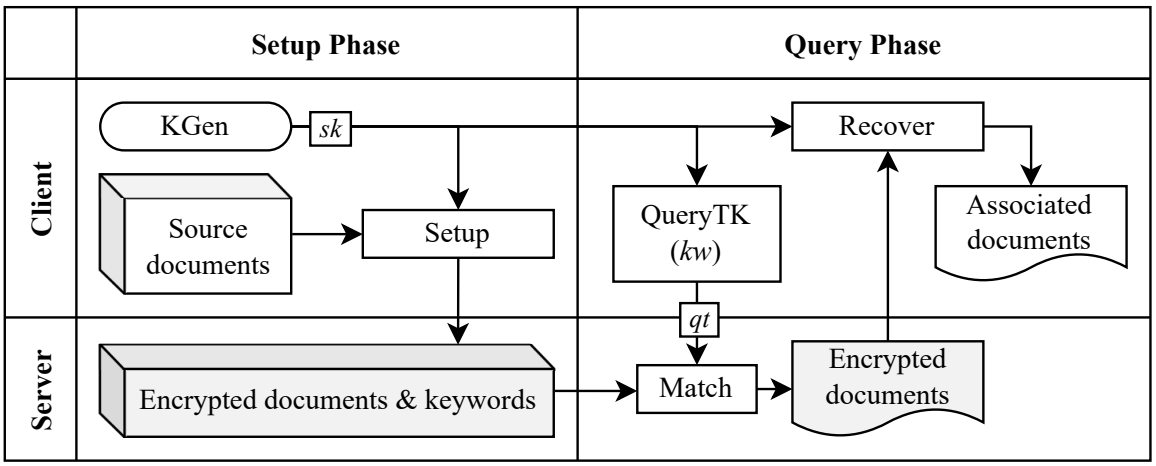


Figure 1. System model of EDESE.

In the setup phase, the client first runs the key generation algorithm, **KGen**, to produce a secret key sk, which is kept confidential. Subsequently, the client executes the setup algorithm, **Setup**, using a collection of source documents as input. This process generates a collection of encrypted documents, which are then sent to and stored by the server.

During the query phase, the client encrypts a queried keyword kw into a query token qt using the query token generation algorithm, **QueryTK**, together with the secret key sk. The query token is sent to the server, which uses the matching algorithm, **Match**, to identify all encrypted documents related to qt and returns them to the client. Finally, the client applies the recovery algorithm, **Recover**, with the secret key sk as input to

retrieve all documents associated with the queried keyword from the received encrypted documents.

4.3.3 Algorithms of EDESE. An easily deployable and efficiently searchable encryption (EDESE) Σ_{EDESE}=(**KGen, Setup, QueryTk, Match, Recover**) is composed of five polynomial-time algorithms as below:

- **KGen**$(1^{\lambda}) \rightarrow \boldsymbol{sk}$: This probabilistic algorithm, run by the client, takes as input a security parameter λ and outputs a secret key sk.
- **Setup**$(\boldsymbol{sk}, \boldsymbol{D}) \rightarrow \boldsymbol{ED}$: It takes as inputs a secret key sk and a set of documents D, and outputs a set of encrypted documents ED combined with the encrypted keywords. It is a probabilistic algorithm run by the client who outsources ED to a server.
- **QueryTK**$(\boldsymbol{sk}, \boldsymbol{kw}) \rightarrow \boldsymbol{qt}$: This deterministic algorithm takes as inputs the secret key sk and a keyword kw, and outputs a query token qt. This algorithm is run by the client.
- **Match**$(\boldsymbol{ED}, \boldsymbol{qt}) \rightarrow \boldsymbol{ct}$: It takes as inputs the encrypted documents ED and a query token qt, and outputs a response ct, which is a subset of ED. This algorithm is run by the server.
- **Recover**$(\boldsymbol{sk}, \boldsymbol{ct}) \rightarrow \boldsymbol{pt}$: It takes as inputs the secret key sk and the response ct, and outputs a set of documents pt. This algorithm is run by the client.

Correctness. For any kw, sk, and D, it calculates $ED \leftarrow$ **Setup**(sk, D), $qt \leftarrow$ **QueryTK**(sk, kw), $ct \leftarrow$ **Match**(ED, qt), and $pt \leftarrow$ **Recover**(sk, ct). Then, a Σ_{EDESE} is **correct** if and only if pt is identical to the search result $D(kw)$, which indicates the set of documents related to the keyword kw in D.

4.3.4 LAAs on EDESE.

In recent years, numerous studies have demonstrated that underlying keywords of query tokens can be retrieved by exploiting the vulnerabilities of EDESE schemes, given prior knowledge about the targeted database, like Blackstone et al. (2019); Cash et al. (2015, 2013); Gui et al. (2023); Islam et al. (2012); Ning et al. (2021). This prior knowledge includes either a partially or fully encrypted database being targeted, along with auxiliary information. Specifically, the targeted encrypted database is obtained from the server in the setup phase or observed via communications between the client and the server during the query phase.

Attack models can be categorized into two types based on the extent of knowledge contained in the auxiliary information: known-data attacks and inference attacks. In a stronger attack model like Blackstone et al. (2019), known as the known-data attack, the auxiliary information is partially or the entire underlying plaintext of the targeted encrypted database. In a weaker attack model, known as the inference attack, the auxiliary information is a similar database that has the same or close distribution as the targeted database.

In general, the attacker learns the leakage patterns, including access pattern, search pattern, volume pattern, co-occurrence pattern, etc., from the targeted encrypted database and the auxiliary information via leakage functions, which we will formalize the commonly used leakage functions at the end of this subsection. The attacker's objective is to establish the mapping relationships between the query tokens and the keywords.

In this paper, we consider a strong LAA model on EDESE, characterized by the complete observation of the targeted encrypted database and utilizing the entire underlying plaintext database as auxiliary information. This attack model encompasses both the setup phase and the query phase, as the target encrypted database is acquired

through these stages. It is worth noting that during the query phase, certain dummy information can be filtered out.

This paper focuses solely on addressing volume leakage, leaving aside search, access, and co-occurrence patterns. This choice aligns with the nature of EDESE, as methods to mitigate these other patterns would undermine its efficiency and easy deployment. For example, ORAM-based searchable encryption can hide multiple patterns but introduces significant communication overhead, compromising EDESE's efficiency. Similarly, randomized query token generation could help with search and access pattern leakage, but requires additional mapping and mathematical computations, which counteract the ease of deployment of EDESE. Moreover, preventing volume pattern leakage alone is adequate to protect EDESE against LAAs introduced inBlackstone et al. (2019); Cash et al. (2015, 2013); Gui et al. (2023); Islam et al. (2012); Ning et al. (2021).

Leakage Functions. To better demonstrate leakage functions, our common math functions are defined, including: 1) $Eq(a, b) = 1$ if $a = b$; otherwise, $Eq(a, b) = 0$; 2) given a matrix $M_{m\times n}$, $Row(M, i) = (M_{i,1}, ..., M_{i,n})$; 3) $Col(M, i) = (M_{1,i}, ..., M_{m,i})$; 4) and given a vector or an array V, $|V|$ counts the number of elements in V.

– **Search pattern leakage function.** The search pattern leakage function $\mathcal{L}_S$ identifies whether two queries qt_1, qt_2 refer to the same keyword, i.e. $\mathcal{L}_S(qt_1, qt_2) = Eq(qt_1, qt_2)$.

– **Access pattern leakage function.** The access pattern leakage function $\mathcal{L}_A$ produces a matrix $M^{(m,n)}$, reflecting the relationships between a set of n encrypted documents ED and a set of m query tokens Q, such that $M_{i,j} = 1$ if $qt_i \in Vec(ED, ed_j)$; otherwise $M_{i,j} = 0$, where $qt_i \in Q, ed_j \in ED, i \in [1, m], j \in [1, n]$.

– **Volume pattern leakage function.** These functions capture the number of associations in the access pattern matrix M. The query token volume leakage function $\mathcal{L}_{QV}$ (response volume) computes the number of encrypted documents linked to each query token: $\mathcal{L}_{QV}(qt_i) = |Row(M,i)|$, where $i \in [1,m], qt_i \in Q$. Similarly, the document volume leakage function $\mathcal{L}_{DV}$ outputs the number of query tokens linked to each encrypted document: $\mathcal{L}_{DV}(ed_j) = |Col(M,j)|$, where $j \in [1,n], ed_j \in ED$.

– **Co-occurrence pattern leakage function.** These functions measure overlap between query tokens or documents. The query token co-occurrence leakage $\mathcal{L}_{QC}$ yields a matrix $M^{(QC)} \in \mathbb{Z}^{m\times m}$ where for all $j_1, j_2 \in [1,n]$, $M^{(QC)}_{i_1,i_2} = |Row(M,i_1) \cap Row(M,i_2)|$. The document co-occurrence leakage $\mathcal{L}_{DC}$ similarly produces $M^{(DC)} \in \mathbb{Z}^{n\times n}$, with $M^{(DC)}_{j_1,j_2} = |Col(M,j_1) \cap Col(M,j_2)|$.

4.3.5 Leakage-Resilient EDESE.

We demonstrate the Leakage-Resilient EDESE (LR-EDESE) with k-indistinguishability and leakage function $\mathcal{L}$ in Definition 4.

Definition 4 (k-ind $\mathcal{L}$-resilient EDESE)**.** *Let* $\Sigma = (\mathbf{KGen}, \mathbf{Setup}, \mathbf{QueryTK}, \mathbf{Match}, \mathbf{Final})$ *be an EDESE scheme,* $\lambda \in \mathbb{N}$ *be the security parameter,* k *be a threshold number,* $\mathcal{L}$ *be a polynomial time leakage function,* $\mathcal{A}$ *be a probabilistic polynomial time adversary, and* $\mathcal{C}$ *be a polynomial time challenger with the response capability of the following oracle.*

– $\mathcal{O}_{\mathbf{QTGen}}(\cdots)$*: $\mathcal{A}$ submits a keyword $kw_i \in K(D)$, where $i \in [1, |K(D)|]$.*
$\mathcal{C}$ returns a deterministic query token $qt_{(i,\alpha')}$, where $\mathcal{L}(qt_{(i,1)}, ED) = \cdots = \mathcal{L}(qt_{(i,m)}, ED) = \mathcal{L}(qt_i, ED)$, $\alpha' = \alpha + l \mod m$, m indicates the total number of query tokens satisfying the equivalent leakage to qt_i, $m \geqslant k$, $qt_{(i,\alpha)} = qt_i$, and $qt_i \leftarrow \mathbf{QueryTK}(sk, kw_i)$.

We consider the following experiment $\mathbf{LR}^{\Sigma}_{\mathcal{A}}(\lambda, k, \mathcal{L})$*:*

Experiment $\mathbf{LR}^{\Sigma}_{\mathcal{A}}(\lambda, k, \mathcal{L})$***:***

$sk \leftarrow \mathbf{KGen}(1^{\lambda}), \quad l \in_R [0, k-1]$

$ED \leftarrow \mathbf{Setup}(sk, D)$

where $\mathcal{O}_{\boldsymbol{QTGen}}(\cdot)$ *on input* kw_i :

if $i \notin [1, |K(D)|]$, ***return*** $\perp$

else $i \leftarrow i + 1$, ***return*** $qt_{(i,\alpha')}$

$l' \leftarrow \mathcal{A}^{\mathcal{O}_{\boldsymbol{QTGen}}}(D, ED)$

if $l' = l$, ***return*** 1

else, return 0

We say that Σ *is a* k*-ind* $\mathcal{L}$*-resilient EDESE if for all probabilistic polynomial time* $\mathcal{A}$*,*

$$\mathbf{Adv}^{\mathbf{LR}}_{\mathcal{A}}(\lambda, k, \mathcal{L}) = \left| \mathbb{P}[\mathbf{LR}^{\Sigma}_{\mathcal{A}}(\lambda, k, \mathcal{L}) = 1] - \frac{1}{k} \right| \leqslant negl(\lambda).$$

For Definition 4, the challenger $\mathcal{C}$ runs **Setup** algorithm to produce obfuscated and encrypted documents ED based on the documents D generated by $\mathcal{A}$. Then, $\mathcal{C}$ chooses a random number l from 0 to $k - 1$. In the query phase, $\mathcal{A}$ queries the oracle $\mathcal{O}_{\boldsymbol{QTGen}}$ multiple times by all keywords in D. $\mathcal{O}_{\boldsymbol{QTGen}}$ will output a query token qt corresponding to the given keyword kw based on the secret key sk, ED, the $\mathbf{QueryTK}$ algorithm, and the chosen number l, such that qt have the same leakage as the query token based on kw. In the guess phase, $\mathcal{A}$ guesses l' for l. If $l = l'$, $\mathcal{A}$ wins the security game, which means $\mathcal{A}$ can distinguish a query token based on a given keyword from a randomly chosen query token when both of them have the same leakage regarding the given leakage function. Otherwise, then we say the EDESE scheme is k-ind $\mathcal{L}$-Resilient secure.

4.4 VLR-EDESE

An EDESE scheme qualifies as a Volume Leakage-Resilient EDESE (VLR-EDESE) when it achieves k-ind $\mathcal{L}_{QV}$ resilient security and k-ind $\mathcal{L}_{DV}$ resilient security

simultaneously. Thus, a VLR-EDESE must integrate both a query volume leakage hiding method and a document volume leakage hiding method.

The remainder of this section introduces a novel document volume leakage hiding method and a query volume leakage hiding method inspired by Bost and Fouque (2017)'s work. We then present a concrete construction of VLR-EDESE along with its security analysis. Furthermore, we discuss padding strategies for optimizing VLR-EDESE's efficiency and security.

4.4.1 Document Volume Hiding Method. A modulo-based partition approach is defined to hide document volume pattern leakages, which achieves the intermediate computational complexity and the intermediate number of obfuscated documents (duplicated documents + dummy documents) compared to 1) no partitioning approach and 2) a flexible partition approach which partitions keywords associated with a document into flexible sized disjoint subsets.

Modulo-Based Document Partition (MBDPar) Algorithm. Given a set of source documents D_n and the set of associated keywords K_m, the algorithm produces a set of obfuscated documents $D'_{n'}$ and the corresponding modulus p^*, which is shown in Algorithm 1.

The algorithm first computes the optimal modulus value p^* such that the total number of result documents is minimal, which is represented by n'_p. Then, the MBDPar algorithm partitions the keyword vector associated with each source document to produce multiple disjoint subsets associated with multiple duplicated documents. There is no more than one sub-vector whose size is less than p^* while the sizes of others are p^*, following the modulo calculation. After that, the MDBPar algorithm associates keywords in each sub-vector with a duplication of the source document.

Algorithm 1: Modulo-based document partition algorithm MBDPar

Require: a set of source documents D_n and the set of corresponding keywords K_m.
Ensure: a set of obfuscated documents $D'_{n'}$, the optimal modulus p^*.

computes the maximum document volume value $v^* = \max_{d \in D_n} Vol(d)$;
for all $p \in [1, v^*]$ **do**
  initialize $O_p \leftarrow \{0\}_p, O'_p \leftarrow \{0\}_p$;
  for all $i \in [1, n]$ **do**
    $O_p[p] \leftarrow O_p[p] + \lfloor \frac{Vol(D,d_i)}{p} \rfloor$;
    if $\gamma > 0$ **then**
      $O_p[\gamma] \leftarrow O_p[\gamma] + 1$, where $\gamma = Vol(D, d_i) \bmod p$;
    end if
  end for
  for all $v \in [1, p]$ **do**
    if $O_p[v] > 0$ **then**
      set $O'_p[v] \leftarrow max(k - O_{p^*}[v], 0)$;
    else
      $O'_p[v] \leftarrow 0$;
    end if
  end for
  compute $n'_p \leftarrow \sum_{v=1}^{p} O_p[v] + O'_p[v]$;
end for
select the p^* such that n'_{p^*} is minimum;
initialize $D' \leftarrow \emptyset$;
for all $d_i \in D_n$ **do**
  compute $u_i \leftarrow \lceil \frac{Vol(D,d_i)}{p^*} \rceil$;
  set $V \leftarrow Vec(D, d_i)$;
  for all $i' \in [1, u_i]$ **do**
    $d_{i,i'} \leftarrow_{duplicate} d_i$;
    if $i' < u_i$ **then**
      update $Vol(D', d_{i,i'}) \leftarrow p^*$;
    else
      $Vol(D', d_{i,i'}) \leftarrow Vol(D, d_i) - (i-1) \times p^*$;
    end if
    update $Vec(D', d_{i,i'})$ with first $Vol(D', d_{i,i'})$ elements of V;
    $V \leftarrow V \setminus Vec(D', d_{i,i'})$;
  end for
end for
for all $v \in [1, p^*]$ **do**
  generate $O'_p[v]$ dummy documents $dd_1, \cdots, dd_{O'_p[v]}$ and updates $Vec(D, *)$ and $Vol(D, *)$ functions for each of them with randomly selected v non-repeated keywords from K_m, then put them into D';
end for
output p^* and $D'_{n'}$ where $n' \leftarrow n'_{p^*}$.

Next, the algorithm counts occurrences of values from 1 to p^* in document volume values. If the occurrence of a value v is less than k and greater than 0, the algorithm creates new dummy documents associated with v keywords randomly selected from K_m to reach k. Finally, it outputs the obfuscated document $D'_{n'}$, which contains all duplicated documents and dummy documents, where n' represents the number of documents in D'.

Computational Complexity and Boundaries. The computational complexity of the MDBPar algorithm is $\mathcal{O}\left(\frac{mn}{p^*} + n + p^*(k-1)\right)$.

The number of obfuscated documents produced by the MBDPar algorithm is bounded by $\mathcal{O}\left(\frac{mn}{p^*} + n + p^*k\right)$. The total number of keyword-document pairs in the obfuscated documents is bounded by $\mathcal{O}(mn + kp^{*2})$.

For further details, refer to 4.6.2.1.

4.4.2 Query Volume Hiding Method. There are many query volume hiding algorithms, such as rounding padding, group padding, and bucket padding approaches. To achieve k-ind query volume resilient secure, recall the definition 4, we need to guarantee each query volume occurs at least k times, which is exactly idea of group padding.

VLR-EDESE employs a group padding algorithm, called keyword clustering algorithm (KClu), inspired by Bost and Fouque (2017) and enhanced with our padding strategy. Given a keyword set K_m, a document set D_n, and a dummy document capacity p^* (i.e., the maximum number of keywords that can be associated with a dummy document), the KClu algorithm outputs a set of obfuscated documents $D'_{n'}$, as described in Algorithm 2.

KClu computes an optimal grouping of keywords to minimize the number of dummy documents needed while ensuring that all keywords within the same group share the same volume. To achieve this, it constructs a weighted directed graph based on sorted keywords where 1) nodes represent keywords plus a start node, 2) edges indicate keyword

Algorithm 2: keyword clustering algorithm KClu

Require: a set of source documents D_n, the set of corresponding keywords K_m, and a dummy document capacity p^*.

Ensure: a set of obfuscated documents $D'_{n'}$.

sort kw in K_m by the ascending order of the $Vol(D_n, kw)$ to obtain K'_m;
construct a weighted directed graph $G = (V, E, W)$, where
$V = [0, m], E = \{(e_1, e_2)|e_1 \in [0, m-k], e_2 \in [e_1 + k, \min(e_1 + 2k - 1, m)]\}$, and the weighting function is defined as
$W(e_1, e_2) = \sum_{e=e_1+1}^{e_2}(Vol(D_n, kw_{e_2}) - Vol(D_n, kw_e))$;
calculate the shortest weighted path P^* from the start vertex 0 to the end vertex m by Dijkstra's Algorithm, where $P^* = \{e_0, e_1, \cdots, e_\mu\}$ and $e_0 = 0, e_\mu = m$;
the optimal collection of keyword groups is generated as $\mathbb{G} = \{g_i\}_{i\in[1,\mu]}$, where $g_i = \{kw_e\}_{e\in[e_{i-1}+1,e_i]}$;
calculate $O[e] = Vol(kw_{e_i}) - Vol(kw_e)$ for $kw_e \in g_i$;
initialize $D' \leftarrow D_n, T \leftarrow \emptyset$;
for all $i \in [1, \mu], e \in [e_{i-1} + 1, e_i]$ **do**
  if $O[e] > |T|$ **then**
    generate total $(O[e] - |T|)$ dummy documents and append them into T and D';
  end if
  select the first $O[e]$ dummy documents from T to form a set T' and update $Vec(D', kw_e) \leftarrow Vec(D, kw_e) \cup T', Vol(D', kw_e) \leftarrow Vol(kw_{e_i})$;
  for all $d \in T'$ **do**
    if $Vol(T', d) = p^*$ **then**
      $T \leftarrow T \setminus \{d\}$;
    end if
  end for
end for
output $D'_{n'}$ where $n' = \sum_{i=1}^{\mu}(e_i - e_{i-1})Vol(kw_{e_i})$.

groups, and 3) the weight function calculates the number of keyword-document pairs required to be padded for a group. The algorithm then applies Dijkstra's shortest path algorithm to identify the optimal grouping P^*. Each group contains at least k and at most $2k-1$ keywords, leading to a total of $(m-2k)k + \frac{k(k-1)}{2}$ edges in the graph.

Once the optimal grouping is determined, KClu pads dummy documents with capacity p^* to form the obfuscated document set $D'_{n'}$, including both dummy and source documents, where n' denotes the total number of obfuscated documents.

Computational Complexity and Boundaries. The computational complexity of the KClu algorithm is $\mathcal{O}\left(m \log m + km + k^2\right)$.

The total number of obfuscated documents generated by the KClu algorithm is $\mathcal{O}(\frac{mn}{p^*} + n)$. Additionally, the number of keyword-document pairs is bounded by $\mathcal{O}(mn)$.

For further details, refer to 4.6.2.2.

4.4.3 Construction of VLR-EDESE. We apply the proposed volume-hiding methods to construct a concrete VLR-EDESE scheme. VLR-EDESE employs a symmetric encryption scheme $\Sigma_{SE} = (\text{KGen}, \text{Enc}, \text{Dec})$ for document encryption and decryption, along with a collision-resistant hash function $\mathcal{H}$ to generate query tokens. The detailed algorithms of VLR-EDESE are presented in Figure 2.

VLR-EDESE utilizes Σ_{SE} to generate the secret key sk, encrypt and decrypt documents, and process query tokens. To mitigate volume pattern leakage, the **Setup** algorithm first applies the MDBPar algorithm to transform the source document set D_n into a semi-obfuscated document set $D'_{n'}$. Next, the KClu algorithm further obfuscates $D'_{n'}$ to produce the final document set $D''_{n''}$. The **Setup** algorithm then encrypts documents in $D''_{n''}$ using Σ_{SE}.Enc and hashes the associated keywords using $\mathcal{H}$.

4.4.4 Padding Strategy. Padding with dummy query tokens becomes ineffective because an attacker can easily filter them out by observing that those dummy

Figure 2. Concrete construction of the VLR-EDESE

- **KGen**$(1^{\lambda}) \rightarrow \boldsymbol{sk}$:
  KGen takes a security parameter λ. Then it runs $sk \leftarrow \Sigma_{SE}.KGen(1^{\lambda})$ and outputs sk.

- **Setup**$(\boldsymbol{sk}, \boldsymbol{D}) \rightarrow \boldsymbol{ED}$ **:**

  1. It initializes $K_m \leftarrow K(D), D_n \leftarrow D$ where $m = |K(D)|, n = |D|$.
  2. Then, for each document $d \in D_n$, it extracts a set of keywords S and updates $Vec(d) \leftarrow S, Vec(kw) \leftarrow Vec(kw) \cup \{d\}, \forall kw \in S$.
  3. Next, it computes $D'_{n'}, p^* \leftarrow \text{MBDPar}(D_n, K_m), D''_{n''} \leftarrow \text{KClu}(D'_{n'}, K_m, p^*)$.
  4. After that, the algorithm encrypts D'' into ED as $ED \leftarrow \{ed|ed = r|\widehat{ed}|qt_1|\cdots|qt_{Vol(d)}, r \xleftarrow{R} \mathbb{R}, \widehat{ed} = \Sigma_{SE}.Enc(sk|r, d), qt_i = \mathcal{H}(sk|kw_i), kw_i \in Vec(d), i \in [1, Vol(d)], \forall d \in D''\}$.

- **QueryTk**$(\boldsymbol{sk}, \boldsymbol{kw}) \rightarrow \boldsymbol{qt}$ **:**
  The algorithm encrypts kw into a query token qt. It calculates $qt = \mathcal{H}(sk|kw)$ and outputs it.

- **Match**$(\boldsymbol{ED}, \boldsymbol{qt}) \rightarrow \boldsymbol{ct}$ **:**
  The server finds the set of encrypted documents ct associated with the given qt from ED.

- **Final**$(\boldsymbol{sk}, \boldsymbol{ct}) \rightarrow \boldsymbol{pt}$ **:**
  The client decrypts all encrypted documents in ct and filters out dummy ones. It initializes $pt \leftarrow \emptyset$. Afterwards, for each $ed \in ct$, it parses ed to $r|\widehat{ed}|qt_1|\cdots|qt_{Vol(d)}$ and then obtain the document as $d = \Sigma_{SE}.Dec(sk|r, \widehat{ed})$. If d is a real document, it put d in pt; otherwise, do nothing. Finally, pt will only hold all documents related to the queried keyword kw.

query tokens are never queried. In addition, padding with neither dummy documents nor dummy query tokens places an additional burden on the client to identify whether a responded document associates with the query token. Taking into account the above reasons, the proposed VLR-EDESE adopts a padding strategy that exclusively utilizes dummy documents, ensuring that only query responses are padded with dummy entries.

4.5 Security Analysis

We prove the security of VLR-EDESE using an expanding strategy. First, we prove that the scheme achieves k-ind $\mathcal{L}_{DV}$ resilience over a set of k documents in Lemma 4.5.1. Next, we extend this result by demonstrating that a collection of such disjoint document sets remains k-ind $\mathcal{L}_{DV}$ resilient secure in Lemma 4.5.2.

Similarly, we prove the k-ind $\mathcal{L}_{QV}$ resilience for query volume leakage. By combining these two results, we establish that the constructed VLR-EDESE satisfies k-ind volume leakage resilient secure, ensuring robust security against LAAs.

Lemma 4.5.1. *Given a set $S^{(ed)}$ of k encrypted documents, it is k-ind $\mathcal{L}_{DV}$ resilient secure if and only if $Vol(ed_1) = Vol(ed_2) = \cdots = Vol(ed_k)$ where $ed_1, ed_2, \cdots, ed_k \in S^{(ed)}$.*

Proof. Since the document volume leakage function $\mathcal{L}_{DV}(d)$ calculates $Vol(ed)$, we have $v^* = \mathcal{L}_{DV}(ed_1) = \mathcal{L}_{DV}(ed_2) = \cdots = \mathcal{L}_{DV}(ed_k)$. Consequently, the adversary $\mathcal{A}$ cannot leverage the leakage function $\mathcal{L}_{DV}$ to extract any information beyond v^*. Given an CPA secure symmetric encryption scheme Σ_{SE}, such as AES-CBC $\mathcal{A}$ cannot distinguish the order of encrypted documents in $S^{(ed)}$. Thus, $S^{(ed)}$ maintains k-ind $\mathcal{L}_{DV}$ resilient security.

Lemma 4.5.2. *Given a collect $\mathbb{C}^{(ed)}$ of encrypted document sets, it is k-ind $\mathcal{L}_{DV}$ resilient secure if and only if $S_i^{(ed)}$ is at least k-ind $\mathcal{L}_{DV}$ resilient secure and $S_i^{(ed)} \cap S_j^{(ed)} = \emptyset$*

where $\forall i, j \in [1, u]$ *with* $i \neq j$ *and* u *indicates the number of encrypted document sets in* $\mathbb{C}^{(ed)}$.

Proof. According to Definition 4, since each set of encrypted documents are k-ind $\mathcal{L}_{DV}$ resilient secure, adversary $\mathcal{A}$ can not identify α produced for $\forall S^{(ed)} \in \mathbb{C}^{(d)}$ with known α'. Therefore, $\mathcal{A}$ has no ability of inferring l for all $S^{(ed)} \in \mathbb{C}^{(d)}$. In such case, the probability to win the security game is $\mathbb{P}[\mathbf{LR}^{\Sigma}_{\mathcal{A}}(\lambda, k, \mathcal{L}_{DV}) = 1] = \mathbf{Adv}^{\mathbf{CPA}_{\mathcal{A},\Sigma_{SE}}}(\lambda) + \frac{1}{k}$. Then, $\mathbf{Adv}^{\mathbf{LR}}_{\mathcal{A}}(\lambda, k, \mathcal{L}_{DV}) \leqslant negl(\lambda)$.

Corollary 4.5.2.1. *If a set of encrypted documents can be divided into a number of disjoint subsets and each subset is* k*-ind* $\mathcal{L}_{DV}$ *resilient secure, then this set of document is* k*-ind* $\mathcal{L}_{DV}$ *resilient secure.*

Proof. Suppose a set $\hat{S}^{(ed)}$ of encrypted documents can be divided into a collect $\mathbb{C}^{(ed)}$ of disjoint subsets, such that $\hat{S}^{(ed)} = \mathbb{U}_{i\in[1,u]} S_i^{(ed)}$ and $S_i^{(ed)} \cap S_j^{(ed)} = \emptyset$ where $\forall i, j \in [1, u]$with $i \neq j$. According to Lemma 4.5.2, if $\forall S^{(ed)} \in \mathbb{C}^{(ed)}$ is k-ind $\mathcal{L}_{DV}$ resilient secure, $\mathbb{C}^{(ed)}$ is k-ind $\mathcal{L}_{DV}$ resilient secure. Consequently, $\hat{S}^{(ed)}$ is also k-ind $\mathcal{L}_{DV}$ resilient secure.

Corollary 4.5.2.2. *If a set of query tokens can be divided in to a number of disjoint subsets and each subset is* k*-ind* $\mathcal{L}_{QV}$ *resilient secure, then this set of query tokens is* k*-ind* $\mathcal{L}_{QV}$ *resilient secure.*

The Corollary 4.5.2.2 can be proved following the same proof processing. Note that k-ind $\mathcal{L}_{QV}$ resilient security relies on cryptographic hash function, which is modeled as a random oracle. This means that the hash function is assumed to provide unpredictable and consistent outputs.

Theorem 4.5.3. *The constructed VLR-EDESE scheme is* k*-ind volume leakage resilient secure.*

Proof. Recall the construction of VLR-EDESE in Figure 2, MDBPar algorithm outputs the semi-obfuscated documents $D'_{n'}$, such that each document volume value occurs at least k times (described in Algorithm 1). Therefore, $D'_{n'}$ is k-ind $\mathcal{L}_{DV}$ resilient secure.

After that, KClu algorithm produces the obfuscated documents $D''_{n''}$ based on $D'_{n'}$. KClu algorithm only adds new dummy documents associated with existing query tokens, therefore, no new document volume values emerges (limited by the modulus p as Algorithm 2). Consequently, $D''_{n''}$ is also k-ind $\mathcal{L}_{DV}$ resilient secure.

In addition, KClu algorithm produces $D''_{n''}$ such that each keyword volume value occurs at least k times. Therefore, $D''_{n''}$ is k-ind $\mathcal{L}_{QV}$ resilient secure.

Since $D''_{n''}$ is k-ind $\mathcal{L}_{DV}$ resilient secure and k-ind $\mathcal{L}_{QV}$ resilient secure, the encryption of $D''_{n''}$ is k-ind volume leakage resilient secure. Hence, the proposed VLR-EDESE is a volume leakage-resilient EDESE with k indistinguishability.

4.6 Evaluations

4.6.1 Overhead of EDESE.

There are three critical overheads of EDESE, including setup computational overhead, server storage overhead, and communication overhead.

– **Setup computational Overhead.** In EDESEs, computational overhead refers to the computational costs associated with the setup phase. It is determined by the computational complexity of the **KGen** and **Setup** algorithms.

– **Server Storage Overhead.** In EDESEs, server storage overhead indicates the data size of encrypted documents. In this paper, we use the number of encrypted documents instead.

– **Communication Overheads.** The communication overheads include client communication (query) overhead and server communication (response) overhead. For EDESE, client communication overhead is represented by the average length of a search request. Server communication overhead is represented by the average associated encrypted documents of a search request. It can be calculated by (the number of document-query_token mappings)/(the number of query tokens).

4.6.2 Theoretical Evaluation. We analyze the complexities of the proposed VLR-EDESE in terms of computational, storage, and communication overheads in Section 4.6.1. The setup computation complexity is $\mathcal{O}\left(m \log m + mn + km + k^2\right)$. The server storage complexity is $\mathcal{O}(km + n)$. The client communication complexity remains constant at $\mathcal{O}(1)$, while the server communication complexity is $\mathcal{O}(km + n)$.

4.6.2.1 MDBPar Algorithm. **Computational Complexity.** The computational complexity of the MDBPar algorithm includes two procedures, optimizing the modulus and padding.

The optimization process consists of two nested iterations, in a computational complexity of $\mathcal{O}(v^* n)$, which is bounded by $\mathcal{O}(mn)$. The padding process firstly partitions document volumes and update Vec, Vol functions for them. The complexity of this process is given by $\mathcal{O}\left(\sum_{i=1}^{n} \left\lceil \frac{Vol(d_i)}{p^*} \right\rceil\right)$. Since $\left\lceil \frac{Vol(d_i)}{p^*} \right\rceil \leqslant \frac{v^*}{p^*} + 1$, we can further bound the complexity as $\mathcal{O}\left(\sum_{i=1}^{n} \left\lceil \frac{Vol(d_i)}{p^*} \right\rceil\right) \leqslant \mathcal{O}\left(\frac{mn}{p^*} + n\right)$. Additionally, the padding process includes the addition of dummy documents and Vec, Vol function updating with a computational complexity of $\mathcal{O}\left(\sum_{v=1}^{p^*} O'_{p^*}[v]\right)$. Recall that the calculation is given by $O'_{p^*}[v] \leftarrow max(k - O_{p^*}[v], 0)$, which ensures that $0 \leqslant O'_{p^*}[v] \leqslant k - 1$. As a result, the computational complexity of adding dummy documents is $\mathcal{O}\left(\sum_{v=1}^{p^*} O'_{p^*}[v]\right) \leqslant \mathcal{O}(p^*(k-1))$.

Combining these computational complexities, the computational complexity of MDBPar is $\mathcal{O}\left(\frac{mn}{p^*} + n + p^*(k-1)\right)$.

Boundaries. The number of obfuscated documents produced by the MBDPar algorithm is proportional to the number of computations required for updating the Vec and Vol functions, the complexity of which was previously calculated as $\sum_{i=1}^{n} \left\lceil \frac{Vol(d_i)}{p^*} \right\rceil + \sum_{v=1}^{p^*} O'_{p^*}[v]$. This is bounded by $\mathcal{O}\left(\frac{mn}{p^*} + n + p^*k\right)$. Additionally, The total number of keyword-document pairs in the obfuscated documents is given by $\sum_{i=1}^{n} Vol(d_i) + \sum_{v=1}^{p^*} vO'_{p^*}[v]$, which is bounded by $\mathcal{O}(mn + kp^{*2})$.

4.6.2.2 KClu Algorithm. Computational Complexity.

The computational complexity of the KClu algorithm includes two procedures, choosing optimal groups forming and padding. The KClu algorithm utilizes Dijkstra's algorithm to find the optimal groups, whose complexity is $\mathcal{O}\left((m+1)\log(m+1) + (m-2k)k + \frac{k(k-1)}{2}\right) \leqslant \mathcal{O}(m\log m + km + k^2)$ and is not effected by the parameter p^*. Then, the algorithm adds dummy documents and updates Vec, Vol functions for each keywords. The number of updates is m.

In a conclusion, the computational complexity of the KClu algorithm is calculated as $\mathcal{O}\left(m\log m + km + k^2\right)$.

Boundaries. The total number of obfuscated documents generated by the KClu algorithm can be determined by dividing the total number of keyword-document pairs in obfuscated documents by the capacity of a dummy document p^*, expressed as $\mathcal{O}(\frac{mn}{p^*} + n)$. Besides, the number of keyword-document pairs is bounded by $\mathcal{O}(mn)$.

4.6.3 Experimental Evaluation.

In our experiments, we utilize the Enron email corpus introduced by Shetty and Adibi (2004), following the methodology of previous studies like Cash et al. (2015); Ning et al. (2021); Pouliot and Wright (2016); Y. Zhang, Katz, and Papamanthou (2016). This dataset comprises 37,470 emails

exchanged by 150 employees of the Enron Corporation between 2000 and 2002. To preprocess the data, we adopt the approach used in Cash et al. (2015); Ning et al. (2021); Y. Zhang et al. (2016), extracting the top 5,000 keywords after removing stop words. We then implement VLR-EDESE using these 5,000 keywords and the complete set of 37,470 emails as source documents.

Evaluation with Varied Parameters. We evaluate the prototype implementation of the proposed VLR-EDESE using varying percentages of source documents and varying values of k. The percentage of source documents ranges at 0.1%, 0.5%, 1%, 5%, 10%, 50%, and 100%. Similarly, the k values are set to 10, 20, 50, and 100. Our evaluation focuses on three key aspects: computation overhead, server storage overhead, and communication overhead.

Table 4 and Figure 3 show that VLR-EDESE is particularly effective for large-scale datasets, especially when k is not too large. Specifically, as highlighted in the blue row of Table 4, for $k \in \{10, 20, 50, 100\}$ and the completed dataset (37,470 documents), VLR-EDESE achieves the following.

- Storage overhead (number of documents) is limited to at most 2 times the source dataset size.
- Communication overhead (number of indices) does not exceed 2.5 times the number of source indices.
- The computational overhead (run-time) is approximately 1 hour.
- Memory consumption does not exceed 8 GB, fitting comfortably within the typical 16 GB RAM of modern PCs.

Furthermore, as demonstrated by the trend lines in Figures 3a and 3b, **the communication and storage overheads of VLR-EDESE decrease as the number**

of source documents increases. Meanwhile, Figures 3c and 3d show that runtime and memory requirements scale almost linearly with the number of source documents.

Moreover, Figures 3e, 3f, and 3h indicate that the rates of obfuscated keyword-document pairs (resp. obfuscated documents) relative to their original counterparts, as well as memory usage, exhibit a linear relationship with k. Finally, Figure 3g confirms that the value of k does not impact the runtime of the setup phase.

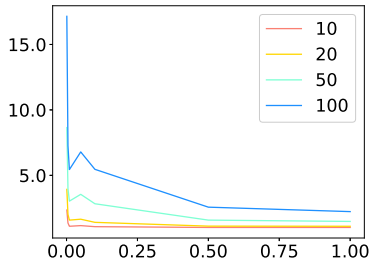
The flexibility of VLR-EDESE allows for tunable security parameters that balance privacy and efficiency. By selecting a moderate k value (e.g., 100), overheads can be significantly reduced. This adaptability enables VLR-EDESE to surpass other schemes in providing adjustable security guarantees alongside optimized storage, communication, and computational efficiency.

Table 4. Performance of the proposed VLR-EDESE scheme.

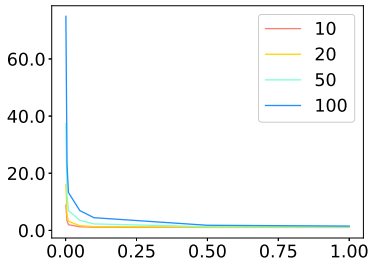
Data	**No. of Indices**[1]					**No. of Documents**				
Pct.	**Source**[4]	10	20	50	100	**Source**	10	20	50	100
0.001	2246	5286 (2.35)	8826 (3.93)	19438 (8.65)	38522 (17.15)	37	328 (8.86)	594 (16.05)	1384 (37.41)	2772 (74.92)
0.005	11995	16350 (1.36)	24207 (2.02)	47817 (3.99)	87173 (7.27)	187	613 (3.28)	1020 (5.45)	2251 (12.04)	4321 (23.11)
0.01	19761	21899 (1.11)	31133 (1.58)	59789 (3.03)	107504 (5.44)	374	745 (1.99)	1193 (3.19)	2600 (6.95)	4961 (13.26)
0.05	112242	130253 (1.16)	185181 (1.65)	398322 (3.55)	762503 (6.79)	1873	2369 (1.26)	3071 (1.64)	6603 (3.53)	12991 (6.94)
0.1	221540	239690 (1.08)	313172 (1.41)	627262 (2.83)	1209040 (5.46)	3747	4224 (1.13)	4955 (1.32)	8513 (2.27)	16638 (4.44)
0.5	1102810	1127398 (1.02)	1235006 (1.12)	1744237 (1.58)	2831602 (2.57)	18735	19498 (1.04)	20409 (1.09)	24442 (1.3)	33467 (1.79)
1	2193545	2245102 (1.02)	2437223 (1.11)	3253320 (1.48)	4901463 (2.23)	37470	38411 (1.03)	39776 (1.06)	45263 (1.21)	56648 (1.51)

Data	**Memory(GB)**[2]					**Runtime(s)**[3]				
Pct.	-	10	20	50	100	-	10	20	50	100
0.001	-	0.01	0.02	0.05	0.09	-	24.5	23.74	22.36	20.15
0.005	-	0.09	0.09	0.18	0.35	-	336.42	338.12	332.03	317.61
0.01	-	0.35	0.35	0.35	0.47	-	564.96	561.03	545.09	529.66
0.05	-	0.47	0.47	0.86	1.72	-	1454.34	1514.67	1450.9	1421.78
0.1	-	1.72	1.72	1.72	2.17	-	1676.92	1670.76	1655.18	1613.32
0.5	-	2.84	3.06	3.49	4.43	-	2927.11	2906.1	2828.16	2858.69
1	-	5.81	5.96	6.53	7.79	-	3743.62	3752.6	3696.84	3661.42

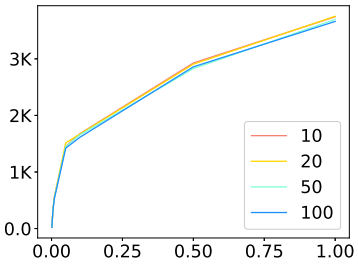
[1] "Indices" indicates keyword-document pairs.
[2] Peak memory consumption (GB).
[3] Setup runtime of VLR-EDESE (seconds).
[4] Original dataset without hiding.
[5] $k \in \{10, 20, 50, 100\}$.



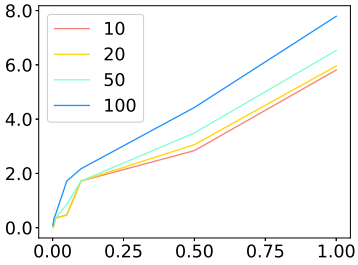
(a) Rates of keyword-document pairs to ones in the baseline EDESE.



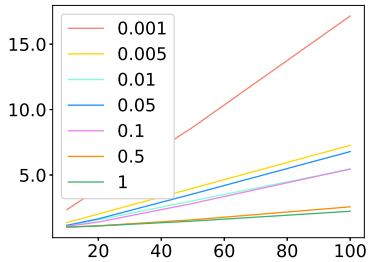
(b) Rates of the obfuscated documents to the source documents.



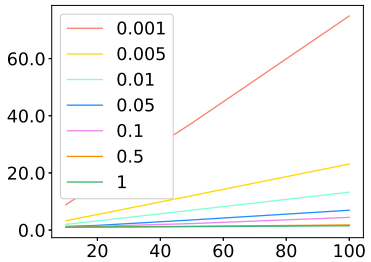
(c) Run times.



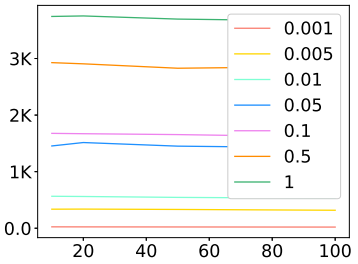
(d) Memories.



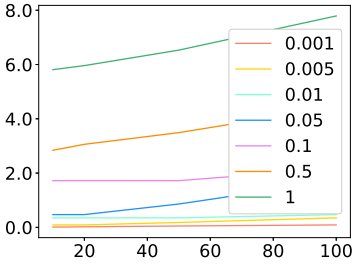
(e) Rates of keyword-document pairs to ones in the Baseline EDESE.



(f) Rates of the obfuscated documents to the source documents.



(g) Run times.



(h) Memories.

Figure 3. Experiment performances in the number of keyword-document pairs, the number of obfuscated documents, setup running times, and memory peaks setup required. (a, b, c, d) indicate performance over percentage numbers of source documents. (e, f, g, h) demonstrate performances over k values.

Comparison with Existing Schemes. This section presents a comparative analysis of the VLR-EDESE against two state-of-the-art SSE schemes, PRT-EMM introduced by Kamara and Moataz (2019) and FP-EMM introduced by Patel et al. (2019), along with the baseline EDESE scheme. To adapt these SSE schemes to the EDESE setting and preserve their security guarantees within the EDESE framework, we apply a straightforward transformation: replacing encrypted indices in each scheme with encrypted documents, ensuring that identical encrypted indices map to the same encrypted document. This adaptation does not account for additional computational overhead incurred on the server. Besides, the baseline EDESE scheme is the EDESE without any leakage protection.

As shown in Table 5, **VLR-EDESE outperforms PRT-EMM and FP-EMM in terms of overall efficiency**, which encompasses server storage, query communication, and server response overheads, as well as setup complexity. Compared to PRT-EMM and FP-EMM, while VLR-EDESE incurs a storage overhead of at most 138× compared to the baseline EDESE, it significantly reduces setup costs and achieves superior efficiency in query processing. Specifically, VLR-EDESE matches the baseline EDESE in client query communication overhead while achieving the lowest server response overhead, equivalent to that of PRT-EMM.

The compared schemes are implemented as follows.

– **Baseline EDESE.** The baseline EDESE encrypts each document along with its associated query tokens without incorporating dummy or duplicate documents. This scheme serves as a baseline for evaluating the efficiency of other schemes.

– **PRT-EMM.** We adopt the most storage-efficient parameter configuration from the original PRT-EMM work with $\alpha = 0.5$.

– **FP-EMM.** FP-EMM employs a full padding strategy, ensuring that all query response lengths are padded to the maximum observed query response length. Each query requires two token transmissions—one for each table—while the server responds twice, returning the maximum possible document set.

Both PRT-EMM and FP-EMM achieve k'-query volume RL security, where $k' = 5000$. To ensure a fair comparison, VLR-EDESE is configured to provide the same k'-indistinguishability in both query volume and document volume pattern leakage. This k' value represents the theoretical security limit achievable by any EDESE or ORAM-based scheme. Each encrypted document is standardized to a size of 1KB to isolate the impact of document encryption.

Table 5. Comparisons of state-of-the-art SSE schemes and the VLR-EDESE scheme.

Scheme	Setup Complexity	Storage (Server)	Query Communication		Overall*
			Client (query)	Server (response)	
Baseline	$\mathcal{O}(mn)$	36.6 MB (1x)	32 B (1x)	438 KB (1x)	1
PRT-EMM	$\mathcal{O}(vn^2)$	30.7 GB (860x)	1.6 MB (50Kx)	22 MB (50x)	320
FP-EMM	$\mathcal{O}(mn)$	3.3 GB (94x)	3 KB (99x)	42 MB (99x)	97
VLR-EDESE	$\mathcal{O}\left(m \log m + mn + km + k^2\right)$	5.1 GB (138x)	32 B (1x)	22 MB (50x)	63

overall rate = (server storage rate + client query rate + server response rate) / 3.

4.7 Application: CloudSec

In this section, we introduce CloudSec,a real-world application that integrates the prototype implementation of VLR-EDESE, enabling secure keyword searches on popular cloud storage services.

4.7.1 System Design. CloudSec consists of three modules, including the user interface management module (UIM module), the cloud service module (CS module), and the encryption module, as illustrated in Figure 4.

– **UIM module** manages interactions between UIs, including the cloud platform management UI, file upload UI, search UI, file viewer, and menu UI. Specifically, the cloud platform management UI allows users to configure connections to

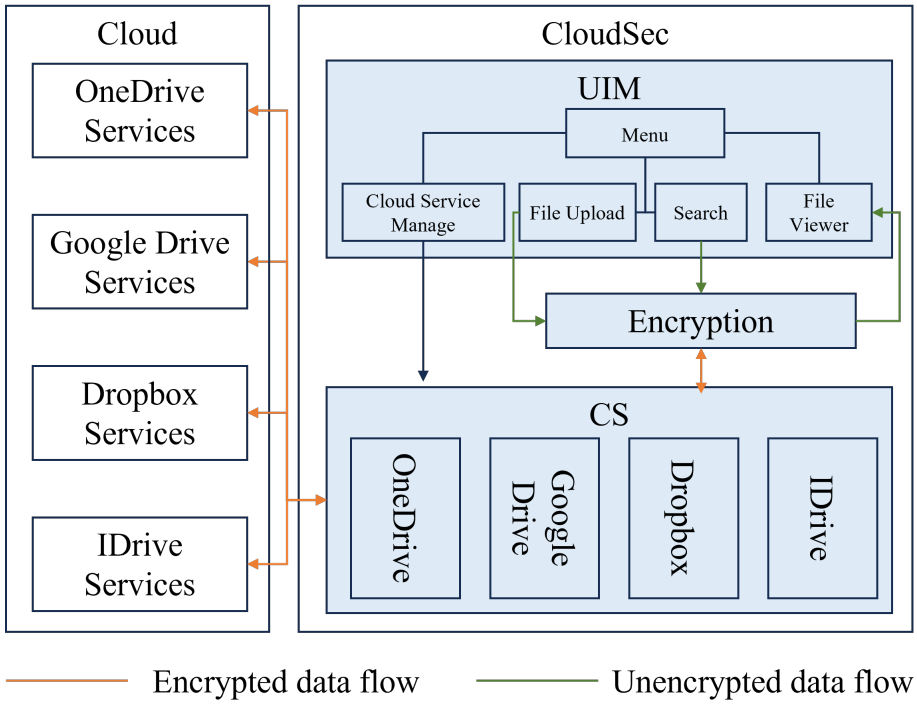


Figure 4. The architecture of CloudSec.

cloud storage services. The file upload UI enables users to upload directories to a specified cloud storage platform. The search UI receives search keyword and selected storage platforms from users. The file viewer displays files from the search result. The menu UI provides access to all functionalities.

– **CS module** handles cloud storage interactions, including user authentication and file operations with the cooperation of the cloud service platform. File operations include uploading, keyword search, and downloading.

– **Encryption module** integrates the prototype implementation of VLR-EDESE and manages user secret keys, ensuring secure keyword search functionality.

4.7.2 User Workflow. Similar to EDESE, CloudSec operates in two phases: setup and query. The workflow is illustrated with OneDrive.

In setup phase, when launching CloudSec, the encryption module runs `KGen` to generate and store the user's secret key. Before uploading files, the user authorizes CloudSec to access OneDrive. The UIM module redirects the "Add Cloud $\rightarrow$ OneDrive" action to the CS module, which invokes OneDrive's authentication. After the user grants access, the CS module stores the resulting token. The user selects a folder, the encryption module encrypts the files, and the CS module uploads them using the stored token.

In query phase, the user inputs a keyword via the UIM search box and selects OneDrive. The encryption module generates a query token, which the CS module sends along with the stored token to OneDrive's search interface. Upon receiving results, the CS module downloads the files via returned URLs. The encryption module then decrypts and filters the files before displaying them in File Explorer.

4.7.3 Implementation. This section outlines key details of the CloudSec implementation. We developed CloudSec in C# for the Windows platform, integrating OneDrive as a cloud storage service. CloudSec interacts with OneDrive via its REST API for seamless file management. To enhance the user experience, CloudSec is designed as a tray application, enabling convenient menu access and robust hotkey functionality for streamlined operations.

Encryption module. The encryption module of CloudSec implements VLR-EDESE using AES-CBC-256 for symmetric encryption and HMAC-SHA-256 for query token generation. Specifically, AES-CBC-256 ensures secure encryption and decryption of documents, while HMAC-SHA-256functions as a keyed hash mechanism for generating query tokens. In addition, a cryptographically secure random number generator (RNG) is employed for secret key generation. All cryptographic operations utilize the System.Security.Cryptography namespace in C#.

The encryption module encrypts both file names and content to ensure data confidentiality. Consequently, all files stored on cloud platforms retain encrypted filenames and contents. To optimize efficiency, the module decrypts only file names to filter out dummy and duplicate entries, eliminating unnecessary file content decryption.

UIM. The UIM module manages the menu UI, cloud service management UI, file upload UI, and search UI, ensuring seamless user interaction. Specifically, the menu UI, accessible via the tray application's right-click menu, provides options for cloud service management, file upload, search, and exit. The cloud service management UI allows users to add or remove connections to cloud storage platforms. The file upload UI includes cloud storage platform selection and a directory selection dialog for specifying files to upload. The search UI presents a search box containing a text input field, checkboxes for selecting target cloud platforms, and a submission button for executing searches.

CS. The CS module facilitates cloud storage integration by utilizing REST APIs, which employ standard HTTP methods for resource operations. Each API request is stateless, meaning it contains all necessary information for execution, as the server does not retain client context between requests. REST APIs are widely used in cloud service development, e.g., OneDrive, Dropbox, IDrive, and Google Drive.

For OneDrive integration, CloudSec employs OneDrive REST APIs with JSON as the data exchange format, which is efficiently handled in C# using Json.NET. The CS module requests OneDrive user authentication through the HTTP URL: login.microsoftonline.com/common/oauth2/v2.0/authorize.
For small file (¡4 MB) uploading, the CS module sends request to [PUT /me/drive/items/parent-id:/filename:/content] while [POST /me/drive/items/itemId/createUploadSession] for large file uploading. The searching REST API is [GET /me/drive/root/search(q='query-

token')]. By leveraging these APIs, the CS module enables secure authentication, efficient file management, and keyword search functionality within CloudSec.

4.7.4 Achievements. In this section, we outline the key achievements of **CloudSec** in terms of security, usability, and flexibility.

– **Security.** As an application of VLR-EDESE, CloudSec preserves the same level of security, specifically volume leakage-resilient EDESE. In alignment with the EDESE design, CloudSec only outsources encrypted files and encrypted file names to cloud storage platforms. All encryption and decryption processes are confined to the local environment, ensuring data confidentiality.

– **Usability.** CloudSec offers a seamless user experience of secure keyword searches on cloud storage platforms. Firstly, the application presents a search textbox, which is similar to existing file search tools, such as Everything and Listary, and minimizes the user learning curve. Then, files of the search result are presented in a File Explorer window as if searching on plaintext files, providing a seamless and intuitive experience for users.

– **Flexibility.** CloudSec's design prioritizes adaptability and ease of integration with existing systems. On one hand, the CS module is capable of connecting with popular cloud storage platforms through REST APIs, enabling it to work seamlessly with current cloud services without complex modifications. On the other hand, following the VLR-EDESE model, CloudSec requires only basic operations—uploading, downloading, and searching—on cloud storage platforms. No additional platform-specific configurations are needed, simplifying implementation and ensuring compatibility with all existing cloud storage providers.

4.7.4.1 Search Result Filtering.

CloudSec generates random names for dummy files and identical names for duplicated files for easy distinguishing. It is unnecessary to distinguish duplicated file names from normal file names. The reason behind is that there are no overlaps between keyword sets of any two duplicated documents. In such case, no duplicated file names appear in one query response. To distinguish dummy files from benign ones, the names of dummy files will start with 0 value and then are followed by , which definitely does not occur at the beginning of benign ones.

4.8 Conclusion

In this chapter, we introduced a novel leakage resilience security concept for EDESE, designed to prevent leakage abuse attacks while preserving the key advantages of EDESE, considering the trade-off between storage and communication overheads. Then, we proposed a novel document volume hiding approach called modulo-based document partitioning and integrated it with a group padding approach for query document volume hiding. Using these methods, we constructed VLR-EDESE, a volume-leakage-resilient EDESE. We have implemented the prototype of VLR-EDESE and conducted experiments to evaluate its performance. Our experiments and evaluations demonstrated the suitability of VLR-EDESE for large-scale EDESE by effectively balancing privacy with computation, storage, and communication overheads, achieving flexible k indistinguishability. Furthermore, we integrated the prototype into CloudSec for real-world use, enabling secure keyword searches on popular cloud storage services, such as OneDrive, Google Drive, Dropbox, and IDrive, via REST APIs.

VLR-EDESE effectively handles volume leakage; however, its symmetric nature limits its flexibility in expressive multi-user scenarios, a challenge we will address in Chapter V through registration-based asymmetric encryption.

CHAPTER V

PEKS IN THE DECENTRALIZED TRUST SETTING: REGISTRATION-BASED ENCRYPTION WITH KEYWORD SEARCH (RBEKS)

This chapter is based on the paper *"RBAE2KS: Registration-Based Authenticated Encryption with Expressive Keyword Search,"* which has been submitted for publication. I was the primary author, responsible for the scheme design, security analysis, implementation, and manuscript preparation. Dr. Yingjiu Li provided conceptual guidance and editorial revisions. Robert H. Deng, Jianting Ning, and Shengmin Xu contributed to the refinement of the manuscript.

While the previous chapter (Chapter IV) established a robust defense against volume leakage in symmetric settings, it inherently relies on a pre-shared secret key, which poses significant key management challenges in large-scale, multi-user environments. Furthermore, most symmetric schemes lack the mathematical structure required to support complex, expressive queries.

Building upon the theoretical shift from fully-trusted to semi-trusted curators discussed in Chapter II, this chapter introduces RBAE2KS. Our goal is to bridge the gap between registration-based encryption and expressive search by utilizing Linear Secret Sharing Schemes (LSSS), effectively eliminating key escrow while providing a high degree of search flexibility.

5.1 Introduction

The rapid proliferation of cloud computing and outsourced data storage has fundamentally reshaped data management, offering unprecedented scalability and accessibility. However, this transformative shift brings paramount concerns regarding data privacy and security. **Public Key Encryption with Keyword Search (PEKS)**, introduced by Boneh et al. (2004a), enables keyword searches directly over encrypted

data in a public-key setting. However, PEKS is vulnerable to **keyword guessing attacks (KGAs)** like Byun, Rhee, Park, and Lee (2006); Huang and Li (2017); Qin, Chen, Huang, Liu, and Zheng (2020); Yau, Heng, and Goi (2008), where malicious actors repeatedly guess keywords, generate corresponding trapdoors, and test matches against intercepted ciphertexts. This brute-force approach allows attackers to deduce original keywords, compromising search query privacy.

To rigorously counteract this vulnerability, **Public Key Authenticated Encryption with Keyword Search (PAEKS)** like K. He, Ma, and Wang (2023); Huang and Li (2017); Noroozi and Eslami (2019); Qin et al. (2020) integrates authentication mechanisms into encryption and search processes, ensuring only legitimately encrypted keywords from authorized senders yield successful matches. This authentication capability prevents adversaries from forging valid ciphertexts or trapdoors, effectively defending against KGAs. **Identity-Based Authenticated Encryption with Keyword Search (IBAEKS)** introduced in Li, Huang, Shen, Yang, and Susilo (2019); Pakniat, Shiraly, and Eslami (2020); Shiraly, Eslami, and Pakniat (2024); Taha et al. (2025) further reduces key management overhead by leveraging Identity-Based Encryption (IBE), eliminating certificate management entirely.

Despite significant security enhancements, current IBAEKS schemes face two critical limitations. First, the inherent **key escrow problem** demonstrated in Al-Riyami and Paterson (2003); Lai, Deng, Zhao, and Weng (2013b): in traditional identity-based cryptosystems, a Private Key Generator (PKG) serves as a central, fully trusted authority holding a master secret key. This grants the PKG the ability to decrypt any ciphertext or generate any user's private key, creating a single point of failure and a high-value target for adversaries. Compromise of the PKG would directly jeopardize the confidentiality of all encrypted data under its purview.

Second, most existing IBAEKS schemes **fail to support expressive queries** like Li et al. (2019); Pakniat et al. (2020); Shiraly et al. (2024); Taha et al. (2025). The majority are restricted to basic keyword matching—often limited to single keywords or simple conjunctive sets. This rigidity severely curtails practical utility in real-world applications demanding sophisticated search capabilities. For instance, searching for documents containing "medical records" while excluding "billing," or requiring "contract" OR "agreement" within particular date ranges—such complex, fine-grained patterns remain largely unaddressed by current IBAEKS.

Interestingly, while numerous **Expressive Asymmetric Searchable Encryption (EASE)** schemes, e.g., Cai, Zhao, Li, Li, and Fan (2025); Cui, Wan, Deng, Wang, and Li (2016); Lai, Zhou, Deng, Li, and Chen (2013); Lv, Hong, Zhang, and Feng (2014); L. Meng, Chen, Tian, Manulis, and Liu (2024b); Shen et al. (2019); Tseng, Fan, and Liu (2022); Q. Xu et al. (2021), have been proposed—some even achieving PAEKS-level security against KGAs—they take a different approach by using public-key and certificate-based frameworks. This design choice trades the key escrow vulnerability for certificate management overhead, reintroducing the complexity that identity-based systems sought to eliminate. Thus, the cryptographic community faces a fundamental tradeoff: existing schemes provide either expressive queries without key escrow (EASE/PAEKS with certificate overhead) or efficient key management (IBAEKS with key escrow and limited expressiveness), but not both desirable properties simultaneously.

These fundamental challenges motivate our pivotal research question: *Can we construct an efficient searchable encryption scheme that simultaneously eliminates key escrow, supports expressive queries, and achieves efficient key management?*

Registration-Based Encryption (RBE) like Cong et al. (2021); Döttling et al. (2023); Fiore et al. (2023); S. Garg et al. (2018); S. Garg, Hajiabadi, Mahmoody,

Rahimi, and Sekar (2019b); Glaeser et al. (2023a); Glaeser, Kolonelos, Malavolta, and Rahimi (2023b); Mahmoody et al. (2022) was recently introduced to address the key escrow problem while retaining the efficient key management of identity-based systems. RBE fundamentally alters the key generation model: instead of a PKG holding a master key, users generate their own public-private key pairs locally and register their public keys with a semi-trusted Key Curator. This approach effectively eliminates traditional key escrow by ensuring no single entity possesses sufficient information to unilaterally reconstruct users' private keys or decrypt ciphertexts.

While several RBE schemes exist, most require non-black-box use of primitives or circuit-specific techniques. For example, GHMR18 introduced by S. Garg et al. (2018), GHMR19 introduced by S. Garg et al. (2019b), and CES21 introduced by Cong et al. (2021) utilize somewhere statistically binding hash (SSBH) functions and hash garbling based on circuits, raising significant application drawbacks. GKMR23 introduced by Glaeser et al. (2023a), DKLL23 introduced by Döttling et al. (2023), and FKP23 introduced by Döttling et al. (2023) introduce RBE based on black-box techniques. However, DKLL23 relies on Learning With Errors (LWE), requiring lattice operations demonstrated in Regev (2009) with $\mathcal{O}(n)$ update overhead per user, where n refers to the user count. GKMR23 and FKP23 utilize vector commitments, achieving better $\mathcal{O}(\sqrt{n})$ update overhead. Specifically, GKMR23 achieves constant ciphertext size while FKP23's ciphertext size is linear in $\sqrt{n}$. Given that expressive searchable encryption ciphertexts can be optimally linear in the keyword count, there is an opportunity to design a key-escrow-free scheme with expressive query support that achieves optimal ciphertext size—linear only in keyword count—by building upon GKMR23. Notably, GKMR23's vector commitments use bilinear maps with prime-order groups, while state-of-the-art expressive searchable encryption schemes, like FEASE introduced by L. Meng et al.

(2024b), use bilinear map-based anonymous Key-Policy Attribute-Based Encryption (A-KP-ABE), suggesting compatibility for achieving expressive search in RBE.

Even with these insights, we cannot trivially use RBE schemes and expressive search schemes as black boxes to construct a solution. This presents three fundamental technical challenges. First, expressive search schemes are typically constructed from A-KP-ABE schemes, which utilize secret sharing techniques such as Linear Secret Sharing Schemes (LSSS) on exponents with recovery via single pairing operations. However, state-of-the-art group-based RBE employs vector commitment techniques requiring two pairing operations, making it difficult to encapsulate and recover secret values on exponents through pairing operations. This necessitates redesigning ciphertext and trapdoor construction to achieve expressive search in RBE by bridging vector commitments and LSSS.

Second, defending against KGAs requires achieving PAEKS through mutual authentication: senders generate ciphertexts using both receivers' public keys and their own secret keys, while receivers generate trapdoors using both their secret keys and target senders' public keys. This pairing mechanism ensures test algorithms execute correctly only when sender and receiver public keys match their respective secret keys. However, since RBE users don't directly obtain others' public keys (they're aggregated in public parameters), it's challenging for senders and receivers to authenticate while eliminating interference from non-relevant aggregated keys. Moreover, PAEKS schemes typically employ asymmetric key generation algorithms for receivers and senders. For example, FEASE receivers maintain two secret values while senders maintain one different secret value. A trivial approach would triple RBE blocks to ensure receivers maintain two secret values and senders maintain one, but reducing this to a single unified building block remains challenging.

To address these challenges, we introduce three novel techniques. First, we develop *hybrid secret sharing* over vector commitments and LSSS. We utilize vector commitment validation equation components as the basis for LSSS operations while hashing keywords into exponents instead of group elements. This enables LSSS operations over vector commitments. In FEASE, LSSS components have keyword hash values as bases; our approach enables canceling these hash values in exponents rather than group elements, as cancellation in group elements during vector commitment calculations is intractable.

Second, we achieve *bilateral randomness splitting* between receivers and senders. We treat RBE's global secret component (analogous to a system master secret key) as part of trapdoor randomness. Secret splitting occurs through two vector commitment validation equations: one powered by sender's secret key for ciphertext generation, one by receiver's secret key for trapdoor generation. Only matched trapdoors and ciphertexts can recover the split secret in the bilinear mapped group exponent.

Third, we design *symmetric operations* in encryption and trapdoor generation such that sender keys and receiver keys share the same structure. Therefore, senders and receivers use identical key generation processes. This enables reusing RBE's efficient single-secret-value design rather than requiring multiple separate secret values per user as in FEASE. This symmetric design maintains RBE's sublinear public parameter size and update efficiency while supporting expressive queries.

5.1.1 Our Contributions. Building upon RBE for eliminating key escrow and EASE's expressive capabilities, our scheme, Registration-Based Authenticated Encryption with Expressive Keyword Search (RBAE^{2}KS), uniquely integrates these paradigms to address pressing challenges in RBE-based expressive searchable encryption. Specifically, RBAE^{2}KS resolves the key escrow issue while providing robust support for

expressive queries. By integrating registration-based key management with advanced search functionalities, our system offers a more secure, flexible, and practical solution for searchable outsourced data.

Our key contributions are:

1. We present **Registration-Based Encryption with Keyword Search (RBEKS)** and **Registration-Based Authenticated Encryption with Keyword Search (RBAEKS)**, the first IBEKS schemes effectively solving the critical key escrow issue in identity-based cryptosystems, serving as the foundation for our research.

2. Our primary achievement extends RBAEKS to **RBAE^{2}KS**—efficient Registration-Based Authenticated Encryption with Expressive Keyword Search, secure under state-of-the-art security models. It achieves no restrictions on keyword or policy sizes and allows arbitrary strings as keywords.

3. All three schemes achieve registration-based key management and maintain state-of-the-art RBE efficiency: sublinear public parameter size, sublinear registration/update efficiency and update counts per user.

4. We formally define a comprehensive security model for RBAE^{2}KS, including robust definitions for keyword security (preventing adversaries from learning keywords from ciphertexts or trapdoors) and trapdoor unlinkability (ensuring adversaries cannot link queries to specific users or prior queries). We provide rigorous security proofs under standard cryptographic assumptions.

5. We provide comprehensive theoretical analysis of computational and communication costs, and demonstrate practical efficiency through extensive experiments on both synthesized and real-world datasets. RBAE^{2}KS achieves strong performance:

encrypting 100 keywords requires ¡1 second, generating a 50-keyword trapdoor takes 0.04 seconds, and testing matches in ¡0.04 seconds—performance comparable to the state-of-the-art PAEKS scheme in L. Meng et al. (2024b).

Chapter Organization. Section 5.2 reviews related work in searchable encryption. Section 5.3 provides necessary cryptographic preliminaries. Section 5.4 details our scheme construction. Section 5.5 presents security analysis. Section 5.6 evaluates performance. Section 5.7 concludes.

5.2 Related Work

Searchable encryption has been a vibrant area of cryptographic research, driven by the increasing need for secure data outsourcing to cloud environments. Our work critically builds upon, and seeks to advance, several foundational and contemporary concepts within this field. Specifically, this section reviews the state of the art in expressive asymmetric searchable encryption, existing techniques for preventing keyword guessing attacks in public key searchable encryption, and the emerging paradigm of registration-based encryption.

5.2.1 Expressive Asymmetric Searchable Encryption (EASE).

The foundational **Public Key Encryption with Keyword Search (PEKS)** schemes, while pioneering, were primarily limited to exact keyword matching, for example, Boneh et al. (2004a). This inherent restriction severely curtails their practical applicability in real-world scenarios that demand more sophisticated search functionalities. This limitation spurred the development of **Expressive Asymmetric Searchable Encryption (EASE)** schemes such as Cai et al. (2025); Cui et al. (2016); Lai, Zhou, et al. (2013); Lv et al. (2014); L. Meng et al. (2024b); Shen et al. (2019); Tseng et al. (2022); Q. Xu et al. (2021), which aim to support a broader spectrum of query types, transcending simple

equality checks. These include, but are not limited to, conjunctive, disjunctive, subset, range, and wildcard searches, providing greater flexibility for data retrieval.

The initial advancements in EASE often drew heavily from the rich theoretical frameworks of Attribute-Based Encryption (ABE), particularly Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) introduced by Agrawal and Chase (2017b); Riepel and Wee (2022b). For instance, early pioneering schemes by L. Meng et al. (2024b) and Shen et al. (2019) explored the integration of ABE-like access structures into searchable encryption. In these constructions, the complex expressive query is typically encapsulated within an access policy, while the user's secret key (or a component derived from it) serves as the trapdoor. The core matching process then ingeniously mimics the decryption procedure characteristic of ABE. Subsequent research has diligently refined these constructions, with a focus on enhancing efficiency, bolstering security against adaptive adversaries, and expanding support for an even wider array of query types. Notable examples include fuzzy keyword search, which tolerates minor errors or variations in keywords like P. Xu, Jin, Wu, and Wang (2012), and monotone span program policy search like Lv et al. (2014). Despite significant progress, the design of truly expressive, efficient, and provably secure EASE schemes remains a formidable challenge, especially when considering strong security requirements such as KGA defense security and practical deployment constraints. It is crucial to note that many existing EASE schemes are built directly on underlying PEKS or Symmetric Searchable Encryption (SSE) primitives and, as such, often inherit their inherent limitations, such as the keyword guessing attack vulnerability (in the case of PEKS-based EASE) or a reliance on a fully trusted central authority.

5.2.2 PEKS against Keyword Guessing Attacks (KGAs). As elucidated in the introduction, conventional PEKS schemes are inherently vulnerable to Keyword

Guessing Attacks (KGAs), wherein an attacker can deduce keywords by repeatedly testing guesses. To effectively counter this critical vulnerability, the cryptographic community has proposed several sophisticated schemes that primarily incorporate robust authentication mechanisms. The most prominent categories of such authenticated searchable encryption schemes include: **Identity-Based Authenticated Encryption with Keyword Search (IBAEKS)**, **Certificate-Based Authenticated Encryption with Keyword Search (CBAEKS)**, and **Certificateless Authenticated Encryption with Keyword Search (CLAEKS)**.

The current research state for each of these categories is detailed below, highlighting their core mechanisms, advantages, and disadvantages:

– **Identity-Based Authenticated Encryption with Keyword Search (IBAEKS):** This paradigm directly integrates the principles of Identity-Based Cryptography (IBC) into PEKS. The sender's identity and corresponding private key are used to authenticate the encrypted keyword, which is then verified during the search operation. Early pioneering works, such as that by Cai et al. (2025); Cui et al. (2016); Lai, Zhou, et al. (2013); Lv et al. (2014); L. Meng et al. (2024b); Shen et al. (2019); Tseng et al. (2022); Q. Xu et al. (2021) , laid the groundwork for IBAEKS, demonstrating its efficacy in preventing KGAs. These schemes frequently leverage bilinear pairings for their underlying cryptographic operations, enabling efficient authentication and search. Many existing IBAEKS constructions achieve robust security, including chosen-ciphertext security for encryption and strong keyword security against KGAs. However, a significant drawback common to almost all existing IBAEKS schemes is the inheritance of the key escrow problem from the underlying IBE primitive. Furthermore, the majority of these constructions are limited to supporting only basic keyword matching functionalities.

– **Certificate-Based Authenticated Encryption with Keyword Search (CBAEKS):** CBAEKS schemes, e.g., Gritti, Susilo, and Plantard (2016); H. Liu et al. (2024); Z.-Y. Liu, Tseng, Tso, Chen, and Mambo (2021); Lu, Li, and Wang (2020); Lu, Li, and Zhang (2019); X. Yang, Chen, Wang, Li, and Wang (2020), represent a hybrid approach, combining aspects of Public Key Infrastructure (PKI) with elements of identity-based cryptography. In this model, public keys are bound to user identities through digital certificates issued and managed by a Certificate Authority (CA). CBAEKS constructions aim to mitigate the full key escrow issue of traditional IBE while retaining some of the simplified key management associated with identity-based systems. Authentication is achieved by leveraging the sender's certified public key, which the receiver or server can verify against the CA's public parameters. However, CBAEKS typically involves more complex key setup and management procedures compared to pure IBE/IBAEKS due to the overhead of certificate issuance, revocation, and validation processes.

– **Certificateless Authenticated Encryption with Keyword Search (CLAEKS):** CLAEKS schemes like Arabnouri and Shafieinejad (2024); D. He, Ma, Zeadally, Kumar, and Liang (2017); X. Liu et al. (2019); Shiraly, Pakniat, Noroozi, and Eslami (2022); Y. Zhang, Wen, Zhang, and Wang (2019) strive to eliminate the shortcomings of both IBE (key escrow) and PKI (certificate management) models. In this setup, a Key Generation Center (KGC) generates only a partial private key for a user, while the user independently generates the remaining component of their private key locally. This ensures that neither the KGC nor the user alone possesses the complete private key. CLAEKS schemes, exemplified by works such as Shiraly et al. (2022), have demonstrated the feasibility of achieving authenticated searchable encryption without either key escrow or certificates. The security analysis for CLAEKS is often

considerably more intricate due to the distributed nature of key generation and the necessity to address a wider range of attack models, including those by the KGC itself or by malicious users. These schemes might also introduce additional communication overhead during the initial key setup phase.

A detailed comparison of these prominent approaches to countering KGAs, highlighting their respective strengths and weaknesses, is provided in Table 6.

Table 6. Comparison of authenticated searchable encryption schemes.

Feature	**IBAEKS**	**CBAEKS**	**CLAEKS**	**RBAE^{2}KS**
Key Management Model	Identity-based (PKG generates private keys)	Certificate-based (CA issues certificates)	Certificateless (KGC generates partial private keys; users generate remainder)	Registration-based (users generate key pairs and register public keys with KC)
Key Escrow	**Yes** (PKG holds master key)	**Mitigated** (CA verifies but doesn't hold master keys)	**No** (no single entity controls private key)	**No** (no single entity controls private key)
KGA Protection	✓	✓	✓	✓
Trust Model	Fully trusted PKG	Trusted CA	Semi-trusted KGC	Semi-trusted KC
Overhead	Simple key setup	Certificate management (issuance, revocation)	Complex key generation; additional communication	Simple key setup
Query Expressiveness	Basic (exact, conjunctive)	Expressive (monotone boolean formula)	Basic (exact, conjunctive)	**Expressive (monotone boolean formula)**

In summary, while IBAEKS, CBAEKS, and CLAEKS all successfully aim to protect PEKS against KGAs through robust authentication mechanisms, they diverge significantly in their underlying key management models and the degree to which they rely on trusted authorities. IBAEKS offers simplicity in key distribution but is burdened by the key escrow problem. CBAEKS mitigates this escrow but introduces the overhead of certificate management. CLAEKS attempts to eliminate both but brings its own complexities in key generation and a shared trust model for private key construction. Our proposed work specifically distinguishes itself by focusing on the **Registration-Based Encryption (RBE)** paradigm as a distinct and strong solution to the key escrow problem, aiming to combine its inherent benefits with the advanced functionality of expressive query support within the IBAEKS framework.

5.2.3 Registration-Based Encryption.

The concept of **Registration-Based Encryption (RBE)** emerged as a direct and innovative response to the inherent key escrow problem universally prevalent in traditional Identity-Based Encryption (IBE) schemes. In classical IBE, a Private Key Generator (PKG) occupies a position of immense power, holding a master secret key from which it derives and issues all users' private keys. This centralized authority, while simplifying public key management by allowing identities to serve as public keys, inherently possesses the capability to decrypt any ciphertext or forge any user's private key within the system. This concentration of power creates a critical single point of failure and represents a significant trust burden for users.

RBE, first formally introduced by S. Garg et al. (2018), fundamentally shifts this centralized trust model. In contrast to a single, fully-trusted PKG generating all private keys, RBE schemes typically involve a semi-trusted entity, often termed a Key Curator (KC). In this groundbreaking paradigm, users generate their own public/private key pairs locally and independently. The primary role of the semi-trusted KC is then to validate and register the users' public keys (and their associated identities) within the system, rather than being involved in the generation of their private keys. This innovative decentralization of the private key generation process is paramount, as it ensures that no single entity, including the RA, possesses sufficient information to unilaterally reconstruct a user's private key or decrypt all ciphertexts. Consequently, RBE effectively eliminates the traditional key escrow vulnerability. Extensive research in RBE has explored various constructions, pursuing different security properties like chosen-ciphertext security and striving to achieve efficiency comparable to traditional IBE. While RBE demonstrably addresses the critical key escrow problem, its integration into more complex cryptographic primitives like searchable encryption, particularly those

demanding support for expressive queries, introduces new and intricate challenges in cryptographic design and rigorous security analysis. Our work explores the application of RBE principles to address the key escrow issue within the context of IBAEKS supporting expressive queries.

5.3 Preliminaries

In this section, we define query structures, monotone span programs for expressive queries, hardness assumptions, partially hidden structures, and the RBAE^{2}KS system model.

5.3.1 Notation. For integers m, n where $m < n$, $[m, n]$ denotes the set $\{m, m+1, \ldots, n\}$. For $m = 1$, we simply write $[n]$. For a prime number p, $\mathbb{Z}_p$ denotes the set $\{0, 1, \ldots, p-1\}$ where addition and multiplication are computed modulo p. The set $\mathbb{Z}_p^*$ is the same as $\mathbb{Z}_p$ except excluding 0. For a set S, $s \leftarrow_R S$ denotes that s is sampled uniformly and independently at random from S. For sets S_1 and S_2, we denote $S_1 \setminus S_2 = \{s \in S_1 : s \notin S_2\}$ as the set difference. For an algorithm $\mathcal{A}$, $y \leftarrow \mathcal{A}(x)$ denotes that y is the output of running algorithm $\mathcal{A}$ on input x. An adversary is a probabilistic algorithm. A probabilistic algorithm is called probabilistic polynomial time (PPT) if its running time is bounded by some polynomial in the length of its input.

We use bold letters to denote vectors and matrices, with the former in lowercase and the latter in uppercase. By default, a vector $\mathbf{v}$ is treated as a column vector. $\mathbf{v}_i$ denotes the i-th element of $\mathbf{v}$ and $||$ denotes concatenation of vectors. For a matrix $\mathbf{M}$, $\mathbf{M}_i$ denotes the i-th row and $\mathbf{M}_{i,j}$ denotes the j-th element of the i-th row.

5.3.2 Bilinear groups and assumptions. Let $\mathbb{G}$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p and g be a generator of $\mathbb{G}$. A bilinear pairing is a mapping $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ that satisfies the following properties:

– **Bilinearity**: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$.

– **Non-degeneracy**: $e(g, g) \neq I$ where I is the identity element of $\mathbb{G}_T$.

– **Computability**: For all $u, v \in \mathbb{G}$, $e(u, v)$ can be efficiently computed.

Assumption 1 (Bilinear Diffie-Hellman)**.** *Let $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$ be a bilinear group generator. We say that the Bilinear Diffie-Hellman problem holds for $\mathcal{G}$ if no probabilistic polynomial time (PPT) adversary can distinguish the distribution $(\mathcal{G}, g^x, g^y, g^z, e(g, g)^{xyz})$ from the tuple $(\mathcal{G}, g^x, g^y, g^z, e(g, g)^{\omega})$ with non-negligible advantage over random guess, where $x, y, z, \omega \leftarrow_R \mathbb{Z}_p$.*

Assumption 2 (n-Power Diffie-Hellman)**.** *Let $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$ be a bilinear group generator. We say that the n-Power Diffie-Hellman problem holds for $\mathcal{G}$ if no probabilistic polynomial time (PPT) adversary can distinguish the following distributions with non-negligible advantage over random guess:*

$$\left(\mathcal{G}, \{g^{x^i}\}_{i \in [2n] \setminus \{n+1\}}, g^y, e(g, g)^{x^{n+1} y}\right)$$

$$\approx \left(\mathcal{G}, \{g^{x^i}\}_{i \in [2n] \setminus \{n+1\}}, g^y, e(g, g)^z\right)$$

where $x, y, z \leftarrow_R \mathbb{Z}_p^*$.

5.3.3 Query structure.

In this paper, we define *access structures* and *attribute sets* in ABE system as *query structures* and *keyword sets* in EASE paradigm. Below we provide only the definition of the latter set of terms.

Definition 5 (Query structure)**.** *If $\mathcal{K}$ denotes the universe of keywords, then a query structure $\mathbb{Q}$ is a collection of non-empty subsets of $\mathcal{K}$, i.e., $\mathbb{Q} \subseteq 2^{\mathcal{K}} \setminus \{\emptyset\}$. It is called monotone if for every $B, C \subseteq \mathcal{K}$ such that $B \subseteq C$, $B \in \mathbb{Q} \Rightarrow C \in \mathbb{Q}$.*

Monotonicity captures the natural idea that if a document acquires more keywords, it cannot lose any matching. A (monotone) Boolean formula consists of **AND**

and **OR** gates, where each input is associated with a keyword in $\mathcal{K}$. A set of keywords $W \subseteq \mathcal{K}$ satisfies a Boolean formula if we set all inputs of the formula that map to a keyword in W to true and the others to false, and the formula evaluates to true.

Monotone span programs (MSP) (or linear secret sharing schemes demonstrated by Beimel (1996b)) are a more general class of functions and include Boolean formulas. We encode a query structure by a keyword policy $(\mathbf{M}, \pi)$ where $\mathbf{M}$ is a matrix of size $m_1 \times m_2$ over $\mathbb{Z}_p$ and $\pi : \{1, \cdots, m_1\} \rightarrow \mathcal{K}$ is a general mapping function. A. B. Lewko and Waters (2011) present a simple and efficient method to convert any (monotone) Boolean formula F into an MSP $(\mathbf{M}, \pi)$ such that every row of $\mathbf{M}$ corresponds to an input in F and the number of columns is the same as the number of **AND** gates in F. Furthermore, each entry in $\mathbf{M}$ is either 0, 1, or -1.

Furthermore, let $S = \{k_i\}_{i \in [m_3]} \subseteq \mathcal{K}$ be a set of m_3 keywords and $I = \{(i, j) | i \in [m_1], j \in [m_3], \pi(i) = k_j\}$ be the mappings between rows of $\mathbf{M}$ and keywords in S. We say that $(\mathbf{M}, \pi)$ accepts S if there exists a linear combination of mappings that gives $(1, 0, \ldots, 0)$. In other words, there exist constants $\gamma_i \in \mathbb{Z}_p$ for $(i, j) \in I$ such that $\sum_{(i,j) \in I} \gamma_i \mathbf{M}_i = (1, 0, \ldots, 0)$. These constants can be computed in time polynomial in the size of $\mathbf{M}$.

5.3.4 Partially hidden structures. We apply the partially hidden structure for a keyword set and a keyword policy in our proposed schemes. This structure was first proposed by Lai, Deng, Guan, and Weng (2013) for an anonymous CP-ABE and then applied in expressive ASE schemes like Cui et al. (2016); R. Meng et al. (2017). Specifically, each keyword is divided into a generic keyword name and a keyword value. The keyword values used in both encryption and key generation are not disclosed to the cloud server, while only the keyword names are disclosed. These keyword names construct a partially hidden keyword policy structure and keyword set in a secret key and

ciphertext, respectively. The decryption algorithm matches the keyword names first and then decrypts the ciphertext by testing if the keyword values match. If not, it repeats to find another set of keyword names until no new set is found or decryption succeeds.

In RBAE^{2}KS, the keyword set is partially hidden after encryption while the keyword policy defined in the query structure is partially hidden after trapdoor generation. More specifically, given a keyword set $S = \{(n_i, v_i)\}_{i\in[m_3]}$ where n_i represents the i-th keyword's name and v_i represents its corresponding keyword value, a partially hidden keyword set in a ciphertext is defined as $S_n = \{n_i\}_{i\in[m_3]}$, containing only keyword names. Similarly, given a query structure $\mathbb{Q} = \big(\mathbf{M}_{m_1\times m_2}, \pi, \{(n_{\pi(i)}, v_{\pi(i)})\}_{i\in[m_1]}\big)$, the corresponding partially hidden query policy structure is structured as $\mathbb{Q}_n = \big(\mathbf{M}_{m_1\times m_2}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]}\big)$. Therefore, neither v_i used in the ciphertext nor $v_{\pi(i)}$ used in the trapdoor is exposed.

Similar to the scheme introduced by L. Meng et al. (2024b), our scheme has the restriction that each keyword name can only be used once in an access policy. We can achieve multiple uses of a keyword name in a partially hidden query structure by applying the generic transformation given in the A. Lewko, Okamoto, Sahai, Takashima, and Waters (2010b)'s work, which does not increase the size of the ciphertext but does increase the key size.

5.3.5 Registration-Based Searchable Encryption Cryptosystem Model.

All our schemes share the same system model, consisting of four entities: the sender, the receiver, the key curator, and the cloud service provider, which are illustrated below:

– **Sender**: The sender is responsible for user key pair generation for itself, helper key updating, and encryption.

– **Receiver**: The receiver is responsible for user key pair generation for itself, helper key updating, and trapdoor generation. Notably, the sender and receiver share the same key pair generation algorithm.

– **Key Curator (KC)**: The KC is a semi-trusted authority responsible for user registration and one-time-trusted system setup.

– **Cloud Service Provider (CSP)**: The CSP is responsible for encrypted data storage and search operations.

The construction of RBE introduced by Glaeser et al. (2023a) is shown in Fig. 5. The sender and receiver first register with the KC to obtain their public keys and secret keys. Then, the sender encrypts documents with the sender's secret key and uploads the encrypted documents to the CSP. When the receiver wants to search for documents, it generates a trapdoor with its secret key and sends it to the CSP. The CSP performs search operations on the encrypted documents with the trapdoor and returns the search results to the receiver.

5.3.6 Registration-Based Encryption (RBE). The syntax of RBE proposed by Glaeser et al. (2023a) is defined below:

Definition 6 (Registration-Based Encryption (RBE))**.** *Registration-based encryption for a universe of identities* $[N]$ *consists of five PPT algorithms (**Setup**, **KGen**, **Register**, **Enc**, **Dec**) working as follows.*

– ***Setup***$(1^\lambda, N) \to \mathtt{crs}$*: The setup algorithm **Setup** samples a common reference string* $\mathtt{crs}$ *once and for all at the beginning of the protocol.*

– ***KGen***$(\mathtt{crs}, u) \to (\mathtt{pk}, \mathtt{sk})$*: The probabilistic algorithm **KGen** takes as input the common reference string* $\mathtt{crs}$ *and a user identity* u *and outputs a pair of public and*

secret keys ($\mathtt{pk}, \mathtt{sk}$). The key generation algorithm is run locally by any honest party who wants to register itself into the system.

- ***Register***$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, u, \mathtt{pk}) \rightarrow \mathtt{pp}'$*: The deterministic algorithm* ***Register*** *takes as input the common reference string $\mathtt{crs}$, the current public parameter $\mathtt{pp}$, a registering identity u, and a public key $\mathtt{pk}$ (supposedly for the identity u), and it outputs $\mathtt{pp}'$ as the updated public parameters. The* ***Register*** *algorithm uses read and write oracle access to the auxiliary information $\mathtt{aux}$, which will be updated to $\mathtt{aux}'$ during the registration process. (The system is initialized with public parameters $\mathtt{pp}$ and auxiliary information $\mathtt{aux}$ set to the empty set $\emptyset$.)*

- ***Update***$^{[\mathtt{aux}]}(\mathtt{pp}, u) \rightarrow \mathtt{hpk}$*: The deterministic algorithm* ***Update*** *takes as input the current parameters $\mathtt{pp}$ stored at the KC and an identity u. It has read-only oracle access to $\mathtt{aux}$ and generates a helper key $\mathtt{hpk}$ that can help u decrypt its messages.*

- ***Enc***$(\mathtt{crs}, \mathtt{pp}, u, m) \rightarrow \mathtt{ct}$*: The probabilistic algorithm* ***Enc*** *takes as input the common reference string $\mathtt{crs}$, a public parameter $\mathtt{pp}$, a user identity u, and a message m and outputs a ciphertext $\mathtt{ct}$.*

- ***Dec***$(\mathtt{crs}, \mathtt{pp}, \mathtt{hpk}, u, \mathtt{ct}) \rightarrow m'$*: The deterministic algorithm* ***Dec*** *takes as input the common reference string $\mathtt{crs}$, the current parameters $\mathtt{pp}$, a helper key $\mathtt{hpk}$, a user identity u, and a ciphertext $\mathtt{ct}$ and outputs a decrypted message m'.*

Definition 7 (Completeness, compactness, and efficiency of RBE)**.** *For any interactive computationally unbounded adversary $\mathcal{A}$ with polynomial query complexity, consider the following game $\mathtt{EXP}_{\mathcal{A}}^{Comp}(\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.*

1. ***Initialization.*** *$\mathcal{C}$ sets $\mathtt{pp} = \emptyset, \mathtt{aux} = \emptyset, \mathtt{hpk} = \perp, D = \emptyset, u^* = \perp, t = 0$, samples $\mathtt{crs} \leftarrow$* ***Setup***(1^λ)*, and sends the sampled $\mathtt{crs}$ to $\mathcal{A}$.*

2. *Until $\mathcal{A}$ terminates (which is at most $\mathtt{poly}(\lambda)$ steps), proceed as follows. At every iteration, $\mathcal{A}$ chooses exactly one of the actions below to be performed.*

   (a) ***Registering new (non-target) identity.*** *$\mathcal{A}$ sends some $u \notin D$ and $\mathtt{pk}$ in the support of the **KGen**($\mathtt{crs}$) algorithm to $\mathcal{C}$. $\mathcal{C}$ registers (u, $\mathtt{pk}$) by letting $\mathtt{pp}:=$ **Register**$^{[\mathtt{aux}]}$($\mathtt{crs}$, $\mathtt{pp}$, u, $\mathtt{pk}$) and $D := D \cup \{u\}$.*

   (b) ***Registering the target identity.*** *If $u^* \neq \perp$, skip this step. Otherwise, $\mathcal{A}$ sends some $u^* \notin D$ to $\mathcal{C}$. $\mathcal{C}$ then samples $(\mathtt{pk}^*, \mathtt{sk}^*) \leftarrow$ **KGen**($\mathtt{crs}$), updates $\mathtt{pp}:=$ **Register**$^{[\mathtt{aux}]}$($\mathtt{crs}$, $\mathtt{pp}$, $u^*, \mathtt{pk}^*$) and $D := D \cup \{u^*\}$.*

   (c) ***Encrypting for the target identity.*** *If $u^* = \perp$, then skip this step. Otherwise, $\mathcal{C}$ sets $t := t + 1$. $\mathcal{A}$ sends some $m_t \in \{0,1\}^*$ to $\mathcal{C}$, who sends back a corresponding ciphertext $\mathtt{ct}_t \leftarrow$ **Enc**$(\mathtt{crs}, \mathtt{pp}, u^*, m_t)$ to $\mathcal{A}$.*

   (d) ***Decrypting by target identity.*** *$\mathcal{A}$ sends $j \in [t]$ to $\mathcal{C}$. $\mathcal{C}$ then obtains the update $\mathtt{hpk}^* \leftarrow$ **Update**$^{[\mathtt{aux}]}(\mathtt{pp}, u^*)$ and lets $m'_j =$ **Dec**$(\mathtt{sk}^*, \mathtt{hpk}^*, \mathtt{ct}_j)$.*

3. *The adversary $\mathcal{A}$ wins the game if there is some $j \in [t]$ for which $m'_j \neq m_j$.*

Let $Q \in \mathtt{poly}$ be an upper bound on the number of queries issued by $\mathcal{A}$ and let $\hat{N} = |D| \leq N$ at the end of execution. We require the following properties to hold.

– ***Completeness.*** *For all computationally unbounded adversaries $\mathcal{A}$,*
$\Pr[\mathcal{A}$ *wins* $\mathtt{EXP}_{\mathcal{A}}^{\mathit{Comp}}(\lambda)] = 0$.

– ***Compactness of public parameters and helper keys.*** *For all queries $q \in [Q]$, let $\mathtt{pp}_q$ be the public parameters after the q-th query. Then $|\mathtt{pp}_q|$ is sublinear in $\hat{N}$. Moreover, for all $u \in D$, the size of the corresponding helper key $\mathtt{hpk}_q$ is also sublinear in $\hat{N}$.*

– ***Efficiency of registration and update runtime.*** *The running time of each invocation of the **Register** and **Update** algorithms is sublinear in $\hat{N}$.*

– ***Efficiency of update count.*** *The total number of invocations of* ***Update*** *for identity* u^* *in Step 2(d) of the game* $\texttt{EXP}_{\mathcal{A}}^{Comp}(\lambda)$ *is sublinear in* $\hat{N}$.

The concrete construction of RBE is demonstrated in Figure 5. All our schemes are designed on top of this RBE construction. Therefore, all of them share the same syntax definitions for the algorithms **Setup**, **KGen**, **Register**, and **Update**.

5.4 Our Schemes

Our research begins with the RBE scheme introduced by Glaeser et al. (2023a), the most efficient RBE construction to date, featuring sublinear update cost per user, constant-time encryption requiring only 1 pairing operation, and constant-time decryption requiring only 2 pairing operations. As illustrated in Figure 6, our design strategy unfolds in three stages. First, we adapt RBE into RBEKS, which supports single-keyword search but lacks KGA resistance. Second, we enhance RBEKS with sender authentication to obtain RBAEKS, achieving KGA resistance. Finally, building upon RBAEKS, we develop RBAE^{2}KS with expressive keyword search capabilities as our primary contribution.

5.4.1 From RBE to RBEKS.

We begin with the construction of RBE with keyword search (RBEKS) from the RBE construction in Figure 5, demonstrated in Figure 7.

The syntax of the RBEKS scheme is presented below. An RBEKS scheme consists of seven algorithms. RBEKS shares the same definitions of algorithms **Setup**, **KGen**, **Register**, and **Update** except that identities in RBEKS are tuples of two integers while identities in RBE are single integers. For simplicity, we omit their definitions.

– **Enc**$(\texttt{crs}, \texttt{pp}, \texttt{u}, w) \rightarrow \texttt{ct}$: The probabilistic algorithm **Enc** takes as input the common reference string $\texttt{crs}$, a public parameter $\texttt{pp}$, a user identity $\texttt{u}$, and a keyword w and outputs a ciphertext $\texttt{ct}$.

Setup$(1^\lambda, n, B) \to \mathtt{crs}$:
Generate a bilinear group $\mathcal{G} := \{p, \mathbb{G}, \mathbb{G}_T, g, e\}$, sample $z \leftarrow_R \mathbb{Z}_p^*$.
Set $h_i = g^{z^i}$ for each $i \in [2n]$, $C_i = g^*$ for each $i \in [B]$, and $\Lambda_{j,0} = g^*$ for each $j \in [N]$, where g^* is the identity of $\mathbb{G}$ and $N = nB$. Then, compute:

$$\mathtt{crs} = (\mathcal{G}, \{h_i\}_{i\in[2n]\setminus\{n+1\}}), \mathtt{pp} = \{C_i\}_{i\in[B]}, \mathtt{aux} = \{L_j | L_j = \{\Lambda_{j,0}\}\}_{j\in[N]}.$$

KGen$(\mathtt{crs}, u) \to (\mathtt{pk}, \mathtt{sk})$:
If $u \notin [N]$, output $\perp$. Otherwise, sample $x_u \leftarrow_R \mathbb{Z}_p^*$ and compute $u' = (u - 1 \mod n) + 1$ and set $\mathtt{pk} = \{pk, \Xi\}$, $\mathtt{sk} = x_u$, where $pk = h_{u'}^{x_u}, \Xi = \{\xi_i\}_{i\in[n]}, \xi_i = h_{u'+n-i+1}^{x_u}$ for each $i \in [n] \setminus \{u'\}, \xi_{u'} = 0$.
Register$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, u, \mathtt{pk}) \to \mathtt{pp}'$:

- Check the correctness of Ξ by computing:

$$\begin{aligned} e(pk, h_n) \stackrel{?}{=} & e(\xi_1, g) \stackrel{?}{=} e(\xi_2, h_1) \stackrel{?}{=} \cdots \\ & \stackrel{?}{=} e(\xi_{u'-1}, h_{u'-2}) \stackrel{?}{=} e(\xi_{u'+1}, h_{u'}) \\ & \stackrel{?}{=} \cdots \stackrel{?}{=} e(\xi_n, h_{n-1}). \end{aligned}$$

- Compute $u' = (u - 1 \mod n) + 1, k = \lceil \frac{u}{n} \rceil$ and set $C_k' := C_k \cdot \mathtt{pk}$ to update $\mathtt{pp}$ while other components of $\mathtt{pp}$ remain the same.

- For each $L_j \in \mathtt{aux}$, where $j \in [(k-1)n + 1, kn] \setminus \{u\}$, parse $L_j = \{\Lambda_{j,1}, \cdots, \Lambda_{j,l}\}$. Compute $\Lambda_{j,l+1} = \Lambda_{j,l} \cdot \xi_{j'}$ and set $L_j = L_j \cup \{\Lambda_{j,l+1}\}$, while the rest remain the same, where $j' = (j - 1 \mod n) + 1$.

Update$^{[\mathtt{aux}]}(u) \to \mathtt{hpk}$:
Parse $\mathtt{aux} = \{L_1, \cdots, L_N\}$ and set $\mathtt{hpk} = L_u$.
Enc$(\mathtt{crs}, \mathtt{pp}, u, m) \to \mathtt{ct}$:

- Compute $u' = (u - 1 \mod n) + 1, k = \lceil \frac{u}{n} \rceil$ and find C_k from $\mathtt{pp}$.

- Sample $a \leftarrow_R \mathbb{Z}_p^*$ and compute:

$$\mathtt{ct} = (ct_0, ct_1, ct_2, ct_3) = (C_k, {C_k}^a, g^a, e(h_{u'}, h_{n+1-u'})^a \cdot m).$$

Dec$(\mathtt{crs}, \mathtt{pp}, \mathtt{sk}, \mathtt{hpk}, u, \mathtt{ct}) \to m'$:

- Compute $u' = (u - 1 \mod n) + 1, k = \lceil \frac{u}{n} \rceil$. Then, parse $\mathtt{hpk} = L_u = \{\Lambda_{u,1}, \cdots, \Lambda_{u,l}\}, \mathtt{pp} = \{C_i\}_{i\in[B]}, \mathtt{sk} = x_u$.

- Find $\alpha \in [l]$ and set ${\Lambda_u}' = \Lambda_{u,\alpha}$, such that $A = B$ where $A = e(C_k, h_{n+1-u'}), B = e(\Lambda_{u,\alpha}, g) \cdot e(h_{u'}^{x_u}, h_{n+1-u'})$. If there is no such α, output $\mathtt{GetUpdate}$.

- Then compute:

$$m' = \frac{ct_3 \cdot e\left((\Lambda_u')^{{x_u}^{-1}}, ct_2\right)}{e\left(h_{n+1-u'}^{{x_u}^{-1}}, ct_1\right)}.$$

Figure 5. RBE scheme construction introduced by Glaeser et al. (2023a).

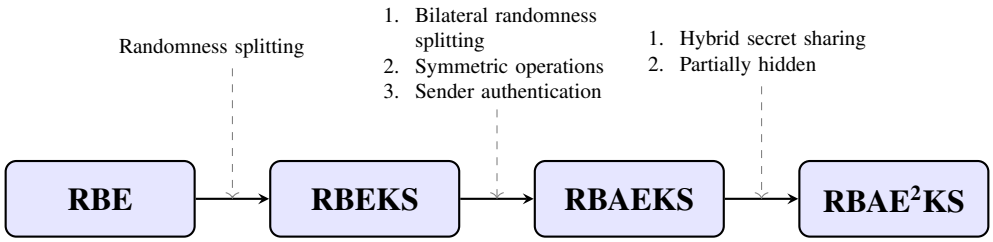


Figure 6. Technical roadmap: evolution from RBE to RBAE^{2}KS.

– **TGen**$(\mathtt{crs}, \mathtt{pp}, \mathtt{hpk}, \mathtt{u}, w) \rightarrow \mathtt{td}$: The probabilistic algorithm **TGen** takes as input the common reference string $\mathtt{crs}$, the current parameters $\mathtt{pp}$, a helper key $\mathtt{hpk}$, a user identity $\mathtt{u}$, and a keyword w and outputs a trapdoor $\mathtt{td}$.

– **Test**$(\mathtt{td}, \mathtt{ct}) \rightarrow 0/1$: The deterministic algorithm **Test** takes as input a trapdoor $\mathtt{td}$ and a ciphertext $\mathtt{ct}$ and outputs 1 if the keywords underlying $\mathtt{td}$ and $\mathtt{ct}$ match; otherwise outputs 0. It is run by the CSP.

To convert RBE to RBEKS, we divide RBE's decryption into two processes: trapdoor generation (which uses secret values) and a test algorithm (which operates without secrets). The generated trapdoor must not reveal the secret key or keyword. To fulfill these requirements, we redesign the encryption algorithm while keeping setup, key generation, registration, and update algorithms unchanged.

Specifically, RBEKS encrypts the value 1 using randomness $a \cdot H(w)$. We package RBE's decryption components as the trapdoor: ${h_{n+1-u'}}^{{x_u}^{-1}}$ and $(\Lambda'_u)^{{x_u}^{-1}}$. However, this trapdoor is not keyword-specific and could decrypt any ciphertext. To bind it to a keyword, we move part of $H(w)$ from ciphertext to trapdoor, yielding $\mathtt{ct} = \left({C_k}^a, g^a, e(h_{u'}, h_{n+1-u'})^{a \cdot H(w)}\right)$ and $\mathtt{td} = \left(h_{n+1-u'}{}^{\frac{H(w)}{x_u}}, (\Lambda'_u)^{\frac{H(w)}{x_u}}\right)$.

We further randomize the trapdoor with $b \in \mathbb{Z}_p^*$, making the exponent $\frac{b \cdot H(w)}{x_u}$. We then rebuild the structures to satisfy the vector commitment validation equation: $\mathtt{ct} = \left({C_k}^a, g^a, {h_{u'}}^{a \cdot H(w)}\right)$ and $\mathtt{td} = \left(h_{n+1-u'}{}^{\frac{b \cdot H(w)}{x_u}}, (\Lambda'_u)^{\frac{b \cdot H(w)}{x_u}}, {h_{n+1-u'}}^b\right)$.

Setup$(1^\lambda, n, B) \to \mathtt{crs}$:
Run $\mathtt{crs} \leftarrow$**RBE.Setup**$(1^\lambda, n, B)$.
KGen$(\mathtt{crs}, \mathtt{u}) \to (\mathtt{pk}, \mathtt{sk})$:
Parse $\mathtt{u} = (u_1, u_2)$ and compute: $\mathtt{pk} = (\mathtt{pk}_1, \mathtt{pk}_2), \mathtt{sk} = (\mathtt{sk}_1, \mathtt{sk}_2)$, where $(\mathtt{pk}_i, \mathtt{sk}_i) \leftarrow$ **RBE.KGen**$(\mathtt{crs}, u_i)$ for $i \in \{1, 2\}$.
Register$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, \mathtt{u}, \mathtt{pk}) \to \mathtt{pp}''$:
Parse $\mathtt{u} = (u_1, u_2)$ and $\mathtt{pk} = (\mathtt{pk}_1, \mathtt{pk}_2)$ and compute: $\mathtt{pp}' \leftarrow$ **RBE.Register**$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, u_1, \mathtt{pk}_1)$, $\mathtt{pp}'' \leftarrow$ **RBE.Register**$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}', u_2, \mathtt{pk}_2)$.
Update$^{[\mathtt{aux}]}(\mathtt{u}) \to \mathtt{hpk}$:
Parse $\mathtt{u} = (u_1, u_2)$ and compute: $\mathtt{hpk} = (\mathtt{hpk}_1, \mathtt{hpk}_2)$, where $\mathtt{hpk}_i \leftarrow$ **Update**$^{[\mathtt{aux}]}(u_i)$ for $i \in \{1, 2\}$.
Enc$(\mathtt{crs}, \mathtt{pp}, \mathtt{u}, w) \to \mathtt{ct}$:

– Parse $\mathtt{u} = (u_1, u_2)$ and compute $u_i' = (u_i - 1 \mod n) + 1, k_i = \lceil \frac{u_i}{n} \rceil$ and find C_{k_i} from $\mathtt{pp}$.

– Sample $a_1, a_2 \leftarrow_R \mathbb{Z}_p^*$ and let $a = a_1 + a_2$. Then compute:

$$\begin{aligned}\mathtt{ct} =& (\{ct_{0,i}, ct_{1,i}, ct_{2,i}\}_{i\in\{1,2\}}, ct_3)\\ =& \left(\{e(C_{k_i}, h_{n+1-u_i'}), {C_{k_i}}^{a_i}, g^{a_i}\}_{i\in\{1,2\}}, {h_{u_2'}}^{a\cdot H(w)}\right).\end{aligned}$$

TGen$(\mathtt{crs}, \mathtt{pp}, \mathtt{sk}, \mathtt{hpk}, \mathtt{u}, w) \to \mathtt{td}$:

– Parse $\mathtt{u} = (u_1, u_2)$ and compute $u_i' = (u_i - 1 \mod n) + 1, k_i = \lceil \frac{u_i}{n} \rceil$. Then, parse $\mathtt{hpk} = (L_{u_1}, L_{u_2}) = (\{\Lambda_{u_1,1}, \cdots, \Lambda_{u_1,l_1}\}, \{\Lambda_{u_2,1}, \cdots, \Lambda_{u_2,l_2}\}), \mathtt{pp} = \{C_i\}_{i\in[B]}, \mathtt{sk} = (x_{u_1}, x_{u_2})$.

– Sample $b \leftarrow_R \mathbb{Z}_p^*$ and compute:

$$\begin{aligned}\mathtt{td} =& (\{td_{0,i}, td_{1,i}, td_{2,i}\}_{i\in\{1,2\}}, td_3)\\ =& \Bigg(\Big\{\underline{\{e(\Lambda_{u_i,j}, g)\cdot e({h_{u_i'}}^{x_{u_i}}, h_{n+1-u_i'})\}_{j\in[l_i]}}, {h_{n+1-u_i'}}^{b\cdot {x_{u_i}}^{-1}\cdot H(w)},\\ & \{{\Lambda_{u_i,j}}^{b\cdot {x_{u_i}}^{-1}\cdot H(w)}\}_{j\in[l_i]}\Big\}_{i\in\{1,2\}}, {h_{n+1-u_2'}}^{b}\Bigg).\end{aligned}$$

Test$(\mathtt{td}, \mathtt{ct}) \to 0/1$:

– Find $\alpha_i \in [l_i]$ such that $td_{0,i,\alpha_i} = ct_{0,i}$, for $i \in \{1, 2\}$. If there is no such α_i, output $\mathtt{GetUpdate}$.

– Compute:

$$A = e(td_3, ct_3), B = \frac{e(td_{1,1}, ct_{1,1})\cdot e(td_{1,2}, ct_{1,2})}{e(td_{2,1,\alpha_1}, ct_{2,1})\cdot e(td_{2,2,\alpha_2}, ct_{2,2})}. \tag{5.1}$$

Then check $A \stackrel{?}{=} B$. If the equation holds, return 1; otherwise, return 0.

Figure 7. Our RBEKS scheme construction.

– The underlined pairing values, e.g., $\underline{\{e(\Lambda_{u_i,j}, g)\cdot e({h_{u_i'}}^{x_{u_i}}, h_{n+1-u_i'})\}_{j\in[l_i]}}$, are computed only once per update. Furthermore, previously computed values can be reused by appending only the new components generated during the $\mathtt{hpk}$ update.

Optimizing Computation. Since the receiver cannot calculate Λ'_u without a ciphertext, the only way to identify the matching Λ'_u is during the test operation. The receiver adds $\{e(\Lambda_{u,i}, g) \cdot e(h_{u'}, h_{n+1-u'})\}_{i\in[l]}$ to the trapdoor, and the sender adds $e(C_{k_r}, h_{n+1-u'})$ to the ciphertext, where $l = |\mathtt{hpk}|$. Crucially, $\{e(\Lambda_{u,i}, g) \cdot e(h_{u'}, h_{n+1-u'})\}_{i\in[l]}$ can be reused after each **Update** by adding only new components, significantly reducing overhead. This *checking design* introduces only l additional pairings for trapdoor generation.

Randomness Splitting Technique. Components in $\mathtt{ct}$ share randomness a and components in $\mathtt{td}$ share randomness b, allowing an attacker to distinguish which keyword was encrypted or queried. Given two keywords w_1, w_2 and a ciphertext $\mathtt{ct} = \left(C_k{}^a, g^a, h_{u'}{}^{a\cdot H(w_\theta)}\right)$ where $\theta \leftarrow_R \{0,1\}$, an attacker can test $e(h_{u'}{}^{a\cdot H(w_\theta)}, h_{n+1-u'}^{x_u}) \stackrel{?}{=} \left(e\left(h_{n+1-u'}, C_k{}^a\right) \cdot e\left((\Lambda_{u,\alpha}), g^a\right)^{-1}\right)^{H(w_{\theta'})}$ to determine if $w_{\theta'}$ was encrypted (i.e., $\theta' = \theta$), where $h_{n+1-u'}^{x_u}$ is observable in user public keys and α is determined by the *checking design*. The trapdoor has a similar vulnerability.

To address this, we adapt the *randomness splitting technique* from the state-of-the-art PAEKS scheme introduced by L. Meng et al. (2024b) to RBE. We split randomness a into $a_1, a_2 \in \mathbb{Z}_p$ with $a = a_1 + a_2$. To encapsulate these values, we define user identity as $\mathtt{u} = (u_1, u_2)$, registering two secret values per user. As shown in Figure 7, ciphertext components become $ct_{1,i} = C_{k_i}^{a_i}$ and $ct_{2,i} = g^{a_i}$. To recover a, trapdoor components are doubled for u_1, u_2 and exponentiated by x_{u_2}, x_{u_1} respectively: $td_{1,i} = h_{n+1-u'_i}{}^{b\cdot {x_{u_i}}^{-1}\cdot H(w)}$ and $td_{2,i} = \{(\Lambda_{u_i,j})^{b\cdot {x_{u_i}}^{-1}\cdot H(w)}\}_{j\in[l_i]}$, where $l_i = |\mathtt{hpk}_i|$.

To maintain correctness while preventing trivial computation of $a_1 + a_2$ via pairings, ct_3 and td_3 remain undoubled: $ct_3 = h_{u'_2}{}^{a\cdot H(w)}$ and $td_3 = h_{n+1-u'_2}{}^{b}$. Now, given $(ct_{1,1}, ct_{1,2}, ct_{2,1}, ct_{2,2}, ct_3)$ or $(td_{1,1}, td_{1,2}, td_{2,1}, td_{2,2}, td_3)$, an attacker observing

$\Lambda_{u_i,\alpha_i}, C_{k_i}, h_{n+1-u'_i}^{x_{u_i}}, h_{u'_i}^{x_{u_i}}$ cannot distinguish keyword w_θ. Both ciphertext and trapdoor now conceal the keyword value.

However, this construction halves RBE block capacity, meaning RBEKS requires double the storage overhead for public parameters to support the same user space.

5.4.2 RBAEKS: Registration-Based Authenticated Encryption with Keyword Search.

We now demonstrate how to convert RBEKS from Section 5.4.1 into RBAEKS.

RBAEKS shares the same **Setup**, **KGen**, **Register**, and **Update** algorithms as RBEKS. For simplicity, we omit their definitions and present only the modified algorithms:

– **Enc**$(\mathtt{crs}, \mathtt{pp}, u_s, \mathtt{hpk}_s, \mathtt{sk}_s, u_r, w) \rightarrow \mathtt{ct}$: The probabilistic algorithm **Enc** takes as input the common reference string $\mathtt{crs}$, a public parameter $\mathtt{pp}$, a sender identity u_s with its helper key $\mathtt{hpk}_s$ and secret key $\mathtt{sk}_s$, a receiver identity u_r, and a keyword w and outputs a ciphertext $\mathtt{ct}$.

– **TGen**$(\mathtt{crs}, \mathtt{pp}, \mathtt{hpk}_r, u_r, \mathtt{sk}_r, u_s, w) \rightarrow \mathtt{td}$: The probabilistic algorithm **TGen** takes as input the common reference string $\mathtt{crs}$, the current parameters $\mathtt{pp}$, a receiver helper key $\mathtt{hpk}_r$, a receiver identity u_r with its secret key $\mathtt{sk}_r$, a sender identity u_s, and a keyword w and outputs a trapdoor $\mathtt{td}$.

– **Test**$(\mathtt{td}, \mathtt{ct}) \rightarrow 0/1$: same as in RBEKS.

RBAEKS resists Keyword Guessing Attacks (KGA), where an attacker (assumed to be the cloud server) generates ciphertexts to test keywords against an existing trapdoor. The scheme achieves:

1. The cloud server cannot match an existing trapdoor by generating a ciphertext.

2. Ciphertext Indistinguishability (CI): ciphertexts do not reveal keyword values.

3. Trapdoor Indistinguishability (TI): trapdoors do not reveal queried keyword values.

Formal definitions of CI and TI are in Section 5.5. In RBAEKS, both sender and receiver have an identity and a public/secret key pair, and register with the key curator to obtain helper keys. Following FEASE's design, *sender authentication* prevents attackers from testing trapdoors through exhaustive encryption: keywords are encrypted using the sender's credentials (u_s, $\mathtt{sk}_s$, $\mathtt{hpk}_s$) and receiver's identity (u_r), while trapdoors use the receiver's credentials (u_r, $\mathtt{sk}_r$, $\mathtt{hpk}_r$) and sender's identity (u_s). Without the sender's secret key, attackers cannot generate valid ciphertexts for a target trapdoor.

Benefiting from RBEKS's double-identity structure, RBAEKS assigns one identity per user but uses two identities (receiver and sender) for operations. This enables *bilateral randomness splitting*: RBAEKS splits randomness a into a_1, a_2 while maintaining single secret keys per user.

We treat RBEKS's identity $\mathtt{u} = (u_1, u_2)$ as receiver identity $u_r = u_1$ and sender identity $u_s = u_2$. We then swap sender's public values and secret values between ciphertext and trapdoor. For example, we switch bases in $ct_{1,2}$ and $td_{1,2}$: $ct_{1,2} = {h_{n+1-u'_s}}^{a_2}$ and $td_{1,2} = {C_{k_s}}^{b \cdot {x_s}^{-1} \cdot H(w)}$. Then we move the sender's secret key from trapdoor to ciphertext: $ct_{1,2} = {h_{n+1-u'_s}}^{a_2 \cdot {x_s}^{-1}}$ and $td_{1,2} = {C_{k_s}}^{b \cdot H(w)}$. Similarly, the remaining components become: $ct_{1,1} = {C_{k_r}}^{a_1}, \quad ct_{2,1} = g^{a_1}, \quad ct_{2,2} = \{{\Lambda_{u_s,i}}^{a_2 \cdot {x_s}^{-1}}\}_{i \in [l_s]} td_{1,1} = {h_{n+1-u'_r}}^{b \cdot {x_r}^{-1} \cdot H(w)}, \quad td_{2,1} = \{{\Lambda_{u_r,i}}^{b \cdot {x_r}^{-1} \cdot H(w)}\}_{i \in [l_r]}, \quad td_{2,2} = g^{b \cdot H(w)}$ where $l_s = |\mathtt{hpk}_s|$ and $l_r = |\mathtt{hpk}_r|$.

Both sender and receiver follow identical key generation and registration algorithms—we call this *symmetric key generation*. This is enabled by randomness splitting, which produces two vector commitment validation equations[1] (for u_r and u_s

[1]For simplicity, we omit values like $H(w)$ and b that do not affect the equations.

respectively), coupled with symmetric operations in ciphertext and trapdoor generation. Unlike traditional PAEKS schemes that use different key generation for sender and receiver, our symmetric approach reduces deployment complexity and improves usability.

The concrete construction of RBAEKS is shown in Figure 8.

5.4.3 RBAE^{2}KS: Registration-Based Authenticated Encryption with Expressive Keyword Search.

As our primary contribution, RBAE^{2}KS extends RBAEKS with expressive query capabilities. Like RBAEKS, it achieves CI and TI to prevent KGA. The scheme consists of seven algorithms are the same as those in RBAEKS except encryption using a set of keywords S and trapdoor generation using a query policy structure $\mathbb{Q}$.

– **Setup**$(1^\lambda, N) \rightarrow \mathtt{crs}$: Samples a common reference string $\mathtt{crs}$.

– **KGen**$(\mathtt{crs}) \rightarrow (\mathtt{pk}, \mathtt{sk})$: Takes as input $\mathtt{crs}$ and outputs a public/secret key pair $(\mathtt{pk}, \mathtt{sk})$. Run locally by any honest party registering in the system.

– **Register**$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, u, \mathtt{pk}) \rightarrow \mathtt{pp}'$: Takes as input $\mathtt{crs}$, current public parameter $\mathtt{pp}$, identity u, and public key $\mathtt{pk}$, and outputs updated public parameters $\mathtt{pp}'$.

– **Update**$^{[\mathtt{aux}]}(\mathtt{pp}, u) \rightarrow \mathtt{hpk}$: Takes as input current parameters $\mathtt{pp}$ and identity u, with *read-only* access to $\mathtt{aux}$, and generates helper key $\mathtt{hpk}$.

– **Enc**$(\mathtt{crs}, \mathtt{pp}, u_s, \mathtt{sk}_s, \mathtt{hpk}_s, u_r, S) \rightarrow \mathtt{ct}$: Takes as input $\mathtt{crs}$, $\mathtt{pp}$, sender credentials $(u_s, \mathtt{sk}_s, \mathtt{hpk}_s)$, receiver identity u_r, and keyword set S, and outputs ciphertext $\mathtt{ct}$.

– **TGen**$(\mathtt{crs}, \mathtt{pp}, u_r, \mathtt{sk}_r, \mathtt{hpk}_r, u_s, \mathbb{Q}) \rightarrow \mathtt{td}$: Takes as input $\mathtt{crs}$, $\mathtt{pp}$, receiver credentials $(u_r, \mathtt{sk}_r, \mathtt{hpk}_r)$, sender identity u_s, and query structure $\mathbb{Q}$, and outputs trapdoor $\mathtt{td}$.

Enc$(\texttt{crs}, \texttt{pp}, u_s, \texttt{hpk}_s, \texttt{sk}_s, u_r, w) \rightarrow \texttt{ct}$:

– Compute $u'_s = (u_s - 1 \mod n) + 1, k_s = \lceil \frac{u_s}{n} \rceil, u'_r = (u_r - 1 \mod n) + 1, k_r = \lceil \frac{u_r}{n} \rceil$.

– Then, parse $\texttt{hpk}_s = L_s = \{\Lambda_{u_s,1}, \cdots, \Lambda_{u_s,l_s}\}, \texttt{pp} = \{C_i\}_{i\in[B]}, \texttt{sk}_s = x_s$, and find C_{k_s}, C_{k_r} from $\texttt{pp}$.

– Sample $a_1, a_2 \leftarrow_R \mathbb{Z}_p^*$ and let $a = a_1 + a_2$. Then compute:

$$\begin{aligned}\texttt{ct} =& (ct_{0,1}, ct_{0,2}, ct_{1,1}, ct_{1,2}, ct_{2,1}, ct_{2,2}, ct_3)\\ =& \Big(e(C_{k_r}, h_{n+1-u'_r}), \underline{\{e(\Lambda_{u_s,i}, g) \cdot e(h_{u'_s}^{x_s}, h_{n+1-u'_s})\}_{i\in[l_s]}}, {C_{k_r}}^{a_1}, {h_{n+1-u'_s}}^{a_2\cdot {x_s}^{-1}},\\ & g^{a_1}, \{{\Lambda_{u_s,i}}^{a_2\cdot {x_s}^{-1}}\}_{i\in[l_s]}, {h_{u'_s}}^{a\cdot H(w)}\Big).\end{aligned}$$

TGen$(\texttt{crs}, \texttt{pp}, u_r, \texttt{hpk}_r, \texttt{sk}_r, u_s, w) \rightarrow \texttt{td}$:

– Compute $u'_s = (u_s - 1 \mod n) + 1, k_s = \lceil \frac{u_s}{n} \rceil, u'_r = (u_r - 1 \mod n) + 1, k_r = \lceil \frac{u_r}{n} \rceil$.

– Then, parse $\texttt{hpk}_r = L_r = \{\Lambda_{u_r,1}, \cdots, \Lambda_{u_r,l_r}\}, \texttt{pp} = \{C_i\}_{i\in[B]}, \texttt{sk}_r = x_r$, and find C_{k_s}, C_{k_r} from $\texttt{pp}$.

– Sample $b \leftarrow_R \mathbb{Z}_p^*$ and compute:

$$\begin{aligned}\texttt{td} =& (td_{0,1}, td_{0,2}, td_{1,1}, td_{1,2}, td_{2,1}, td_{2,2}, td_3)\\ =& \Big(\underline{\{e(\Lambda_{u_r,i}, g) \cdot e(h_{u'_r}^{x_r}, h_{n+1-u'_r})\}_{i\in[l_r]}}, e(C_{k_s}, h_{n+1-u'_s}), {h_{n+1-u'_s}}^{b\cdot {x_r}^{-1}\cdot H(w)},\\ & {C_{k_s}}^{b\cdot H(w)}, \{{\Lambda_{u_r,i}}^{b\cdot {x_r}^{-1}\cdot H(w)}\}_{i\in[l_r]}, g^{b\cdot H(w)}, {h_{n+1-u'_s}}^{b}\Big).\end{aligned}$$

Test$(\texttt{td}, \texttt{ct}) \rightarrow 0/1$:

– Find $\alpha_s \in [l_s], \alpha_r \in [l_r]$ such that $ct_{0,1} = td_{0,1,\alpha_r}, ct_{0,2,\alpha_s} = td_{0,2}$. If there is no such α_i, output $\texttt{GetUpdate}$.

– Check $e(td_3, ct_3) \stackrel{?}{=} \frac{e(td_{1,1}, ct_{1,1})\cdot e(td_{1,2}, ct_{1,2})}{e(td_{2,1,\alpha_r}, ct_{2,1})\cdot e(td_{2,2}, ct_{2,2,\alpha_s})}$. If the equation holds, return 1; otherwise, return 0.

Figure 8. Our RBAEKS scheme construction with the same constructions of algorithms **Setup**, **KGen**, **Register**, and **Update** as those in RBE.

– The underlined pairing values, e.g., $\underline{\{e(\Lambda_{u_s,j}, g) \cdot e({h_{u'_s}}^{x_{u_s}}, h_{n+1-u'_s})\}_{j\in[l_s]}}$ and $\underline{\{e(\Lambda_{u_r,j}, g) \cdot e({h_{u'_r}}^{x_{u_r}}, h_{n+1-u'_r})\}_{j\in[l_r]}}$, are computed only once per update. Furthermore, previously computed values can be reused by appending only the new components generated during the $\texttt{hpk}$ update.

– **Setup**, **KGen**, **Register**, and **Update** algorithms are identical to those defined in RBE in Figure 5.

– **Test**$(\mathtt{ct}, \mathtt{td}) \to 0/1$: Run by the CSP. Outputs 1 if the keyword set in $\mathtt{ct}$ matches the query in $\mathtt{td}$; otherwise 0.

5.4.3.1 Concrete Construction. RBAE^{2}KS builds upon RBAEKS by incorporating Monotone Span Programs (MSP) for expressive queries, using bilinear pairings to achieve both identity-based access control and keyword matching. Figure 9 presents the complete construction.

To integrate MSP into RBAEKS, we require an LSSS recovery equation. Given keywords $S = \{w_i\}_{i\in[m_3]} \subseteq \mathcal{K}$, query structure $\mathbb{Q} = \big(\mathbf{M}_{m_1\times m_2}, \pi, \{k_i\}_{i\in[m_1]}\big)$, and vector $\mathbf{v} = \{\mathbf{v}_i\}_{i\in[m_2]}$, we find subset $K \subseteq S$ accepted by $\mathbb{Q}$ and compute $I = \{(i,j)|i \in [m_1], j \in [m_3], \pi(i) = w_j, w_j \in S\}$ and $\{\gamma_i\}_{(i,j)\in I}$ such that $\sum_{(i,j)\in I}\gamma_i\mathbf{M}_i = (1,\ldots,0)$. The standard LSSS recovery equation is $e(g,g)^{\mathbf{v}_1} = e(\prod_{(i,j)\in I} g^{\mathbf{M}_i\mathbf{v}^\top}, g)^{\gamma_i}$. Since FEASE incorporates keywords into the LSSS equation via KP-ABE, our equation becomes $e(g,g)^{\mathbf{v}_1} = \frac{e\left(\prod_{(i,j)\in I}\left(g^{\mathbf{M}_i\mathbf{v}^\top}\cdot g^{H(\pi(i))}\right), g\right)^{\gamma_i}}{e\left(\prod_{(i,j)\in I} g^{H(w_j)}, g\right)}$. However, combining this with vector commitment validation equations is challenging. We introduce *hybrid secret sharing* to bridge them.

Hybrid Secret Sharing. Observing that $e(g,g)^{\mathbf{v}_1}$ in LSSS resembles $e(h_{u'}, h_{n+1-u'}) = e(g,g)^{z^{n+1}}$ from vector commitment validation, we replace base g in LSSS with $h_{u'}$, yielding $e(g,g)^{\mathbf{v}_1 z^{n+1}} = \frac{e\left(\prod_{(i,j)\in I}\left({h_{u'}}^{\mathbf{M}_i\mathbf{v}^\top}\cdot g^{H(\pi(i))}\right), h_{n+1-u'}\right)^{\gamma_i}}{e\left(\prod_{(i,j)\in I} g^{H(w_j)}, h_{n+1-u'}\right)}$. We construct a vector commitment equation to recover the left side: $\frac{e(C_k, h_{n+1-u'})^{\mathbf{v}_1 x_u^{-1}}}{e(\Lambda_{u,\alpha}, g)^{\mathbf{v}_1 x_u^{-1}}}$. We reuse b from RBAEKS by setting $\mathbf{v}_1 = b$. Since this construction is keyword-independent, it does not leak keyword information. We use the sender's values for the left side (either sender or receiver values suffice).

For the right side, placing $g^{H(w_j)}$ in ciphertext and ${h_{u'}}^{\mathbf{M}_i\mathbf{v}^\top}\cdot g^{H(\pi(i))}$ in trapdoor creates a vulnerability: with public $h_{n+1-u'}$, an attacker can test $e(h_{n+1-u'}, g^{H(w)}) \stackrel{?}{=} e(h_{n+1-u'}, g^{H(w')})$ to guess keywords. We address this by replacing $g^{H(w)}$ with $h_{u'}^{H(w)}$ and constructing $e(g,g)^{z^{n+1}\cdot H(w)}$. Using RBAEKS's two vector commitment equations, we build $e(g,g)^{a\cdot b\cdot z^{n+1}\cdot H(w)}$ (calculated as A in Equation 5.1). Since attackers cannot recover

this value from RBAEKS's ciphertext and trapdoor alone (due to CI and TI security), keyword guessing is prevented.

RBAE^{2}KS uses three vector commitment equations: one recovers the secret value in LSSS, and two jointly cancel attributes. The hybrid validation equation becomes: $\frac{e(C_k,h_{n+1-u'})^{a\cdot b\cdot x_u^{-1}}}{e(\Lambda'_u,g)^{a\cdot b\cdot x_u^{-1}}} = \frac{e\left(\prod_{(i,j)\in I}\left(h_{u'}{}^{\mathbf{M}_i\mathbf{v}^\top}\cdot h_{u'}{}^{H(\pi(i))}\right),h_{n+1-u'}\right)^{a\cdot b\cdot\gamma_i}}{e\left(\prod_{(i,j)\in I} h_{u'}{}^{H(w_j)},h_{n+1-u'}\right)^{a\cdot b}}$.

To simplify, we move all $H(w)$ from RBAEKS trapdoor to ciphertext, ensuring only $h_{u'}{}^{\mathbf{M}_i\mathbf{v}^\top}\cdot h_{u'}{}^{H(\pi(i))}$ involves $H(w)$ in the trapdoor. We remove ct_3, td_3 from RBAEKS so that recovering $e(g,g)^{a\cdot b\cdot z^{n+1}\cdot H(w)}$ requires constructing both vector commitment equations. For efficiency, we compute $h_{u'}{}^{\mathbf{M}_i\mathbf{v}^\top+H(\pi(i))}$ instead of $h_{u'}{}^{\mathbf{M}_i\mathbf{v}^\top}\cdot h_{u'}{}^{H(\pi(i))}$, reducing group exponentiations from 2 to 1 per keyword.

Partially Hidden Structures. Following FEASE, we apply partially hidden structures to eliminate keyword leakage while exposing only keyword names. Each keyword w consists of a name n and value v. The combination $n|v$ is used in cryptographic calculations, but only n appears in ciphertext and trapdoor.

5.5 Security Analysis

In this section, we present the formal security definitions and models for the proposed schemes.

5.5.1 Security Definitions.

5.5.1.1 RBEKS Security.

We adopt the standard Indistinguishability against Chosen Keyword Attacks (IND-CKA) security definition for ASE with registration-based modifications, ensuring ciphertexts reveal no keyword information.

Definition 8 (IND-CKA security)**.** *For any interactive PPT adversary $\mathcal{A}$, consider the game $\mathtt{EXP}_{\mathcal{A}}^{\textit{IND-CKA}}(\lambda)$ between $\mathcal{A}$ and challenger $\mathcal{C}$:*

– ***Setup.*** *$\mathcal{C}$ initializes $\mathtt{pp}=\emptyset, \mathtt{aux}=\emptyset, D=\emptyset, u_r^*=\perp$, samples $\mathtt{crs}\leftarrow$ **Setup**(1^λ), and sends $\mathtt{crs}$ to $\mathcal{A}$.*

Enc$(\texttt{crs}, \texttt{pp}, u_s, \texttt{sk}_s, \texttt{hpk}_s, u_r, S \subseteq \mathcal{K}) \rightarrow \texttt{ct}$:

- Compute $u'_s = (u_s - 1 \mod n) + 1, k_s = \lceil \frac{u_s}{n} \rceil, u'_r = (u_r - 1 \mod n) + 1, k_r = \lceil \frac{u_r}{n} \rceil$.
- Then, parse $\texttt{hpk}_s = L_s = \{\Lambda_{u_s,1}, \cdots, \Lambda_{u_s,l_s}\}, \texttt{pp} = \{C_i\}_{i\in[B]}, \texttt{sk}_s = x_s$, and find C_{k_s}, C_{k_r} from $\texttt{pp}$.
- Let $ct_{0,1} = e(C_{k_r}, h_{n+1-u'_r}), ct_{0,2} = \underline{\{e(\Lambda_{u_s,i}, g) \cdot e(h_{u'_s}^{x_s}, h_{n+1-u'_s})\}_{i\in[l_s]}}$.
- Sample $a_1, a_2, \leftarrow_R \mathbb{Z}_p^*$ and compute $a = a_1 + a_2, ct_{3,1} = h_{n+1-u'_s}{}^{a}, ct_{3,2} = C_{k_s}{}^{a\cdot {x_s}^{-1}}, ct_{3,3} = \{\Lambda_{u_s,i}{}^{a\cdot {x_s}^{-1}}\}_{i\in[l_s]}$. For each keyword $(n_\sigma, v_\sigma) \in S$, where $\sigma \in [|S|]$, compute: $ct_{1,1,\sigma} = C_{k_r}^{a_1\cdot H(n_\sigma|v_\sigma)}, ct_{1,2,\sigma} = h_{n+1-u'_s}^{a_2\cdot H(n_\sigma|v_\sigma)\cdot {x_s}^{-1}}, ct_{2,1,\sigma} = g^{a_1\cdot H(n_\sigma|v_\sigma)}, ct_{2,2,\sigma} = \{\Lambda_{u_s,i}{}^{a_2\cdot H(n_\sigma|v_\sigma)\cdot {x_s}^{-1}}\}_{i\in[l_s]}$.
- Set $\texttt{ct} = \left(S_n, ct_{0,1}, ct_{0,2}, \{ct_{1,1,\sigma}, ct_{1,2,\sigma}, ct_{2,1,\sigma}, ct_{2,2,\sigma}\}_{\sigma\in[|S_n|]}, ct_{3,1}, ct_{3,2}, ct_{3,3}\right)$ where $S_n = \{n_\sigma\}_{\sigma\in[|S|]}$.

TGen$(\texttt{crs}, \texttt{pp}, u_r, \texttt{sk}_r, \texttt{hpk}_r, u_s, \mathbb{Q}) \rightarrow \texttt{td}$:

- Compute $u'_s = (u_s - 1 \mod n) + 1, k_s = \lceil \frac{u_s}{n} \rceil, u'_r = (u_r - 1 \mod n) + 1, k_r = \lceil \frac{u_r}{n} \rceil$.
- Then, parse $\texttt{hpk}_r = L_r = \{\Lambda_{u_r,1}, \cdots, \Lambda_{u_r,l}\}, \texttt{pp} = \{C_i\}_{i\in[B]}, \texttt{sk}_r = x_r$, and $\mathbb{Q} = \left(\mathbf{M}_{m_1\times m_2}, \pi, \{n_{\pi(i)}, \ v_{\pi(i)}\}_{i\in[m_1]}\right)$, and find C_{k_s}, C_{k_r} from $\texttt{pp}$.
- Let $td_{0,1} = \underline{\{e(\Lambda_{u_r,i}, g) \cdot e(h_{u'_r}^{x_r}, h_{n+1-u'_r})\}_{i\in[l_r]}}, td_{0,2} = e(C_{k_s}, h_{n+1-u'_s})$.
- Sample $b \leftarrow_R \mathbb{Z}_p^*, \mathbf{v} \leftarrow_R \mathbb{Z}_p^{*(m_2-1)}$ and let $\hat{\mathbf{v}} = \{b\}||\mathbf{v}$. Then, for each $\mu \in [m_1]$, compute $td_{3,1,\mu} = h_{u'_s}{}^{\mathbf{M}_\mu \mathbf{v}^\top \cdot {x_r}^{-1} + b\cdot H(n_{\pi(\mu)}|v_{\pi(\mu)})}$.
- Compute $td_{3,2} = h_{n+1-u'_s}{}^{b\cdot {x_r}^{-1}}, td_{3,3} = g^{b\cdot {x_r}^{-1}}, td_{1,1} = h_{n+1-u'_r}{}^{b\cdot {x_r}^{-1}}, td_{1,2} = C_{k_s}{}^{b}, td_{2,1} = \{\Lambda_{u_r,i}{}^{b\cdot {x_r}^{-1}}\}_{i\in[l_r]}, td_{2,2} = g^b$.
- Set $\texttt{td} = (\mathbb{Q}_n, td_{0,1}, td_{0,2}, td_{1,1}, td_{1,2}, \ td_{2,1}, td_{2,2}, \{td_{3,1,\mu}\}_{\mu\in[m_1]}, td_{3,2}, td_{3,3})$ where $\mathbb{Q}_n = (\mathbf{M}, \pi, \{n_{\pi(i)}\}_{i\in[m_1]})$.

Test$(\texttt{ct}, \texttt{td}) \rightarrow 0/1$:

- Find $\alpha_s \in [l_s], \alpha_r \in [l_r]$ such that $ct_{0,1} = td_{0,1,\alpha_r}, ct_{0,2,\alpha_s} = td_{0,2}$. If there is no such α_i, output $\texttt{GetUpdate}$.
- Find a set I such that K satisfies $\mathbb{Q}_n$ where $K = \{n_{\pi(i)} | n_j \in S_n, n_{\pi(i)} = n_j\}_{(i,j)\in I}$. Then, find a vector $\{\gamma_i\}_{i\in[|I|]}$ such that $\sum_{(i,j)\in I} \gamma_i \mathbf{M}_i = (1, 0, \cdots, 0)$ where $\mathbf{M}_i$ indicates the i-th row vector. If can not find, return 0.
- Compute $A = e\left(\prod_{(i,j)\in I} td_{3,1,i}^{\gamma_i}, ct_{3,1}\right), B = \frac{e(td_{3,2}, ct_{3,2})}{e(td_{3,3}, ct_{3,3,\alpha_s})}, C_1 = \frac{e\left(td_{1,1}, \prod_{(i,j)\in I} ct_{1,1,j}{}^{\gamma_i}\right)}{e\left(td_{2,1,\alpha_r}, \prod_{(i,j)\in I} ct_{2,1,j}{}^{\gamma_i}\right)}, C_2 = \frac{e\left(td_{1,2}, \prod_{(i,j)\in I} ct_{1,2,j}{}^{\gamma_i}\right)}{e\left(td_{2,2}, \prod_{(i,j)\in I} ct_{2,2,j,\alpha_s}{}^{\gamma_i}\right)}$.
- Check if $A \stackrel{?}{=} B \cdot C_1 \cdot C_2$. If equal, return 1; otherwise, find a new I and repeat the process.
- If no new I can be found, return 0.

Figure 9. Our RBAE^{2}KS scheme construction with the same constructions of algorithms **Setup**, **KGen**, **Register**, and **Update** with those in RBE 5.

- The underlined pairing values, e.g., $\underline{\{e(\Lambda_{u_s,j}, g) \cdot e(h_{u'_s}{}^{x_{u_s}}, h_{n+1-u'_s})\}_{j\in[l_s]}}$ and $\underline{\{e(\Lambda_{u_r,j}, g) \cdot e(h_{u'_r}{}^{x_{u_r}}, h_{n+1-u'_r})\}_{j\in[l_r]}}$, are computed only once per update. Furthermore, previously computed values can be reused by appending only the new components generated during the $\texttt{hpk}$ update.
- **Setup**, **KGen**, **Register**, and **Update** algorithms are identical to those defined in RBE in Figure 5.

$\mathcal{A}$ *performs at most* ***poly***(λ) *registration operations:*

- ***Non-target registration.*** $\mathcal{A}$ *sends* $u \notin D$ *and* $\mathtt{pk}$ *to* $\mathcal{C}$. $\mathcal{C}$ *updates* $\mathtt{pp}$:= ***Register***$^{[\mathtt{aux}]}(\mathtt{crs}, \mathtt{pp}, u, \mathtt{pk})$ *and* $D := D \cup \{u\}$.
- ***Target registration.*** *If* $u_s^* = \perp$, $\mathcal{A}$ *sends* $u_s^* \notin D$. $\mathcal{C}$ *samples* $(\mathtt{pk}_s^*, \mathtt{sk}_s^*) \leftarrow$ ***KGen***$(\mathtt{crs})$, *updates* $\mathtt{pp}$ *and* D, *and returns* $\mathtt{pk}_s^*$.

– ***Phase 1.*** $\mathcal{A}$ *queries trapdoor oracle* $O_T(w)$ *at most* ***poly***(λ) *times.* $\mathcal{C}$ *returns* $\mathtt{td} \leftarrow$ ***TGen***$(\mathtt{crs}, \mathtt{pp}, u_r^*, \mathtt{sk}_r^*, \mathtt{hpk}_r^*, w)$.

– ***Challenge.*** $\mathcal{A}$ *submits* w_0^*, w_1^* *not queried in Phase 1.* $\mathcal{C}$ *returns* $\mathtt{ct}_b^* \leftarrow$ ***Enc***$(\mathtt{crs}, \mathtt{pp}, u_r^*, w_b^*)$ *where* $b \leftarrow_R \{0,1\}$.

– ***Phase 2.*** $\mathcal{A}$ *continues querying* O_T *excluding* w_0^*, w_1^*.

– ***Guess.*** $\mathcal{A}$ *outputs* b' *and wins if* $b = b'$.

An RBEKS scheme is IND-CKA secure if for all PPT adversaries $\mathcal{A}$, $\exists$ *negligible* $\mathbf{negl}$ *such that*

$$Adv_{\mathcal{A}}^{IND-CKA}(\lambda) = \left| Pr\left[b' = b\right] - \frac{1}{2} \right| \leq \mathbf{negl}(\lambda).$$

5.5.1.2 RBAEKS Security.

RBAEKS achieves two security properties: Ciphertext Indistinguishability (CI) and Trapdoor Indistinguishability (TI). We adopt the multi-user IBAEKS security model from , where $\mathcal{A}$ can specify both keywords/policies and sender/receiver identities in oracle queries.

Definition 9 (Ciphertext Indistinguishability (CI))**.** *For any PPT adversary* $\mathcal{A}$, *consider game* $\mathtt{EXP}_{\mathcal{A}}^{CI}(\lambda)$*:*

– ***Setup.*** $\mathcal{C}$ *initializes* $\mathtt{pp} = \emptyset, \mathtt{aux} = \emptyset, D = \emptyset, u_r^* = \perp, u_s^* = \perp$, *samples* $\mathtt{crs} \leftarrow$ ***Setup***(1^λ), *and sends* $\mathtt{crs}$ *to* $\mathcal{A}$.

$\mathcal{A}$ performs registrations as in IND-CKA, additionally registering target receiver u_r^* with $(\mathtt{pk}_r^*, \mathtt{sk}_r^*)$.

– ***Phase 1.*** *$\mathcal{A}$ adaptively queries:*

  * $O_T(u_s, w)$*: Returns* $\mathtt{td} \leftarrow$ ***TGen***$(\mathtt{crs}, \mathtt{pp}, u_r^*, \mathtt{sk}_r^*, \mathtt{hpk}_r^*, u_s, w)$.

  * $O_C(u_r, w)$*: Returns* $\mathtt{ct} \leftarrow$ ***Enc***$(\mathtt{crs}, \mathtt{pp}, u_s^*, \mathtt{sk}_s^*, \mathtt{hpk}_s^*, u_r, w)$.

– ***Challenge.*** *$\mathcal{A}$ submits w_0^*, w_1^* where neither was queried to $O_T(u_s^*, \cdot)$. $\mathcal{C}$ returns $\mathtt{ct}_b^* \leftarrow$* ***Enc***$(\mathtt{crs}, \mathtt{pp}, u_s^*, \mathtt{sk}_s^*, \mathtt{hpk}_s^*, u_r^*, w_b^*)$ *where $b \leftarrow_R \{0, 1\}$.*

– ***Phase 2.*** *$\mathcal{A}$ continues querying with restriction that $O_T(u_s^*, \cdot)$ excludes w_0^*, w_1^*.*

– ***Guess.*** *$\mathcal{A}$ outputs b' and wins if $b = b'$.*

RBAEKS is CI secure if $\forall$ PPT $\mathcal{A}$, $\exists$ negligible $\mathbf{negl}$ *such that $Adv_{\mathcal{A}}^{CI}(\lambda) \leq \mathbf{negl}(\lambda)$.*

Definition 10 (Trapdoor Indistinguishability (TI))**.** *Game $\mathtt{EXP}_{\mathcal{A}}^{TI}(\lambda)$ proceeds identically to $\mathtt{EXP}_{\mathcal{A}}^{CI}(\lambda)$ except:*

– ***Challenge.*** *$\mathcal{A}$ submits w_0^*, w_1^* where neither was queried to $O_C(\cdot, u_r^*)$. $\mathcal{C}$ returns $\mathtt{td}_b^* \leftarrow$* ***TGen***$(\mathtt{crs}, \mathtt{pp}, u_r^*, \mathtt{sk}_r^*, \mathtt{hpk}_r^*, u_s^*, w_b^*)$ *where $b \leftarrow_R \{0, 1\}$.*

– ***Phase 2.*** *$O_C(\cdot, u_r^*)$ queries exclude w_0^*, w_1^*.*

RBAEKS is TI secure if $\forall$ PPT $\mathcal{A}$, $\exists$ negligible $\mathbf{negl}$ *such that $Adv_{\mathcal{A}}^{TI}(\lambda) \leq \mathbf{negl}(\lambda)$.*

5.5.1.3 RBAE²KS Security.

RBAE²KS extends RBAEKS security to expressive queries with partially hidden structures. We adopt the expressive PAEKS model from Cheng and Meng (2022) with: (1) multi-user capability, (2) multi-challenge with keyword sets/policies, and (3) fully-chosen queries for challenge sets/policies.

Definition 11 (Ciphertext Indistinguishability (CI)). *Game* $\texttt{EXP}_{\mathcal{A}}^{CI,ES}(\lambda)$ *extends* $\texttt{EXP}_{\mathcal{A}}^{CI}(\lambda)$ *with:*

– ***Phase 1.*** *$\mathcal{A}$ queries:*

  * $O_T(u_s, \mathbb{Q})$*: Returns* $\texttt{td} \leftarrow$ ***TGen***$(\texttt{crs}, pp, u_r^*, \texttt{sk}_r^*, \texttt{hpk}_r^*, u_s, \mathbb{Q})$.

  * $O_C(u_r, S)$*: Returns* $\texttt{ct} \leftarrow$ ***Enc***$(\texttt{crs}, pp, u_s^*, \texttt{sk}_s^*, \texttt{hpk}_s^*, u_r, S)$.

– ***Challenge.*** *$\mathcal{A}$ submits equal-size sets* $S_0^* = \{n_i, v_{i,0}\}_{i\in[m_3]}, S_1^* = \{n_i, v_{i,1}\}_{i\in[m_3]}$ *sharing names* $\{n_i\}_{i\in[m_3]}$*, where neither satisfies any* $O_T(u_s^*, \cdot)$ *query from Phase 1. $\mathcal{C}$ returns* $\texttt{ct}_b^* \leftarrow$ ***Enc***$(\texttt{crs}, pp, u_s^*, \texttt{sk}_s^*, \texttt{hpk}_s^*, u_r^*, S_b^*)$ *where* $b \leftarrow_R \{0, 1\}$.

– ***Phase 2.*** $O_T(u_s^*, \cdot)$ *queries must not accept* S_0^* *or* S_1^*.

RBAE²KS is CI secure if $\forall$ *PPT $\mathcal{A}$,* $\exists$ *negligible* $\mathbf{negl}$ *such that* $Adv_{\mathcal{A}}^{CI,ES}(\lambda) \leq \mathbf{negl}(\lambda)$.

Definition 12 (Trapdoor Indistinguishability (TI)). *Game* $\texttt{EXP}_{\mathcal{A}}^{TI,ES}(\lambda)$ *extends* $\texttt{EXP}_{\mathcal{A}}^{CI,ES}(\lambda)$ *with:*

– ***Challenge.*** *$\mathcal{A}$ submits equal-size policies* $\mathbb{Q}_0^* = (\boldsymbol{M}^*, \pi^*, \{n_{\pi^*(i)}, v_{\pi^*(i),0}\}_{i\in[m_1]})$ *and* $\mathbb{Q}_1^* = (\boldsymbol{M}^*, \pi^*, \{n_{\pi^*(i)}, v_{\pi^*(i),1}\}_{i\in[m_1]})$ *sharing* $(\boldsymbol{M}^*, \pi^*, \{n_{\pi^*(i)}\}_{i\in[m_1]})$*, where neither accepts any* $O_C(\cdot, u_r^*)$ *query from Phase 1. $\mathcal{C}$ returns* $\texttt{td}_b^* \leftarrow$ ***TGen***$(\texttt{crs}, pp, u_r^*, \texttt{sk}_r^*, \texttt{hpk}_r^*, u_s^*, \mathbb{Q}_b^*)$ *where* $b \leftarrow_R \{0, 1\}$.

– ***Phase 2.*** $O_C(\cdot, u_r^*)$ *queries must not satisfy* $\mathbb{Q}_0^*$ *or* $\mathbb{Q}_1^*$.

RBAE²KS is TI secure if $\forall$ *PPT $\mathcal{A}$,* $\exists$ *negligible* $\mathbf{negl}$ *such that* $Adv_{\mathcal{A}}^{TI,ES}(\lambda) \leq \mathbf{negl}(\lambda)$.

5.5.2 Security Proofs. We prove security under the Generic Group Model (GGM) and random oracle model, consistent with state-of-the-art PEKS schemes (e.g., FEASE).

Theorem 5.5.1. *RBEKS is IND-CKA secure under GGM, the bilinear Diffie-Hellman assumption, and the n-Power Diffie-Hellman assumption, with H modeled as a random oracle.*

Proof. We prove security via game transformations. Let $Adv_{\mathcal{A}}^{(i)} = |\Pr[\mathcal{A} \text{ wins in } Game_i] - 1/2|$.

Game 0. The real IND-CKA game $\texttt{EXP}_{\mathcal{A}}^{\text{IND-CKA}}(\lambda)$ proceeds as follows:

Setup: Challenger $\mathcal{C}$ generates $\texttt{crs} = \{\mathcal{G}, \{g^{z^i}\}_{i\in[2n]\setminus\{n+1\}}\}$ where $\mathcal{G} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$, $z \xleftarrow{\$} \mathbb{Z}_p^*$, and $h_j = g^{z^j}$ for $j \in [n]$.

Registration: Adversary $\mathcal{A}$ registers identities including target $\mathtt{u}_r^* = (u_{r,1}^*, u_{r,2}^*)$. Each identity $\mathtt{u} = (u_1, u_2)$ provides $pk_i = h_{u_i'}^{x_{u_i}}$ and proof Ξ_i satisfying:

$$e(pk_i, h_n) = e(\xi_{i,1}, g) = e(\xi_{i,2}, h_1) = \cdots = e(\xi_{i,n}, h_{n-1}),$$

where ${u_i}' = (u_i - 1 \bmod n) + 1$ for $i \in \{1, 2\}$.

Queries: $\mathcal{A}$ queries trapdoor oracle $O_T(w)$ for $w \notin \{w_0^*, w_1^*\}$, receiving trapdoors with fresh random $b \xleftarrow{\$} \mathbb{Z}_p^*$.

Challenge: $\mathcal{A}$ submits w_0^*, w_1^*. Challenger selects $\theta \xleftarrow{\$} \{0,1\}$, $a_1, a_2 \xleftarrow{\$} \mathbb{Z}_p^*$, and outputs:

$$ct_{0,i} = e(C_{k_i^*}, h_{n+1-{u_i^*}'}), \quad ct_{1,i} = {C_{k_i^*}}^{a_i}, \quad ct_{2,i} = g^{a_i},$$
$$ct_3 = {h_{{u_2^*}'}}^{(a_1+a_2)\cdot t_{w,\theta}^*},$$

where $t_{w,\theta}^* = H(w_\theta^*)$, $C_{k_i^*} = \prod_{\hat{u}_\alpha \in \hat{D}_i} {h_{\hat{u}'_\alpha}}^{x_{\hat{u}_\alpha}}$ with $\hat{D}_i = D_i \setminus \{u_i^*\}$.

Expanding: $ct_{1,2} = {h_{{u_2^*}'}}^{a_2 x_{u_2^*}} \cdot \prod_{\hat{u}_\alpha \in \hat{D}_2} {h_{\hat{u}'_\alpha}}^{a_2 x_{\hat{u}_\alpha}}$.

$\mathcal{A}$ continues queries then outputs θ'. We have $Adv_{\mathcal{A}}^{(0)} = |\Pr[\theta' = \theta] - 1/2|$.

Game 1. Modify challenge: select $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and set:

$$ct_{1,2} = g^{s_1 \cdot x_{u_2^*}} \cdot \prod_{\hat{u}_\alpha \in \hat{D}_2} h_{\hat{u}'_\alpha}{}^{a_2 x_{\hat{u}_\alpha}}, \quad ct_3 = g^{(s_1+s_2)\cdot t^*_{w,\theta}}.$$

Lemma 5.5.2. *If DDH holds in $\mathbb{G}$, then $|Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}| \leq Adv^{DDH}(\lambda)$.*

Proof. Construct PPT algorithm $\mathcal{B}$ receiving DDH challenge $(g, A = g^a, B = g^b, T \in \{g^{ab}, g^r\})$.

Embedding: $\mathcal{B}$ generates `crs` honestly with known z. Set $h_{u_2^{*\prime}} = B$. When u_2^* registers with $pk_2 = B^{x_{u_2^*}}$, verify honestly. Answer all trapdoor queries using known values.

Challenge: Select $\theta \xleftarrow{\$} \{0,1\}$, $a_1 \xleftarrow{\$} \mathbb{Z}_p^*$, set $a_2 = a$. Obtain $t^*_{w,\theta}$ from H. Construct:

$$ct_{1,2} = T^{x_{u_2^*}} \cdot \prod_{\hat{u}_\alpha \in \hat{D}_2} A^{z^{\hat{u}'_\alpha} \cdot x_{\hat{u}_\alpha}},$$

$$ct_3 = B^{a_1 \cdot t^*_{w,\theta}} \cdot T^{t^*_{w,\theta}}.$$

Analysis: If $T = g^{ab}$: $ct_{1,2} = g^{abx_{u_2^*}} \cdot \prod = h_{u_2^{*\prime}}^{a_2 x_{u_2^*}} \cdot \prod$ and $ct_3 = g^{b(a_1+a)t} = h_{u_2^{*\prime}}^{(a_1+a_2)t}$ (Game 0). If $T = g^r$: $ct_{1,2} = g^{rx_{u_2^*}} \cdot \prod$, $ct_3 = g^{(ba_1+r)t}$. Setting $s_1 = r, s_2 = ba_1$ gives Game 1. Thus $Adv^{\text{DDH}}_{\mathcal{B}} = |Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}|$. □

Game 2. Further randomize: select $y \xleftarrow{\$} \mathbb{Z}_p^*$ and set:

$$ct_{1,2} = g^{y} \cdot \prod_{\hat{u}_\alpha \in \hat{D}_2} h_{\hat{u}'_\alpha}{}^{a_2 x_{\hat{u}_\alpha}}, \quad ct_3 = g^{(s_1+s_2)\cdot t^*_{w,\theta}}.$$

Lemma 5.5.3. *If DDH holds, then $|Adv^{(1)}_{\mathcal{A}} - Adv^{(2)}_{\mathcal{A}}| \leq Adv^{DDH}(\lambda)$.*

Proof. Similar DDH reduction: embed challenge in $x_{u_2^*}$ and s_1 relationship. When $T = g^{s_1 x_{u_2^*}}$ (DDH), get Game 1; when $T = g^y$ (random), get Game 2. □

Lemma 5.5.4. $Adv^{(2)}_{\mathcal{A}} = 0$ *for any adversary.*

Proof. In Game 2, only $ct_3 = g^{(s_1+s_2)t^*_{w,\theta}}$ depends on w^*_θ. Since H is a random oracle and w^*_θ is fresh (never queried before challenge or covered by freshness), $t^*_{w,\theta}$ is uniformly random. With uniformly random s_1, s_2, the product $(s_1 + s_2)t^*_{w,\theta}$ is uniformly random, making ct_3 independent of θ. Thus $\Pr[\theta' = \theta] = 1/2$. □

Conclusion.

$$Adv^{\text{IND-CKA}}_{\mathcal{A}}(\lambda) \leq 2 \cdot Adv^{\text{DDH}}(\lambda) = negl(\lambda).$$

□

Theorem 5.5.5. *RBAEKS is CI secure under GGM, the bilinear Diffie-Hellman assumption, and the n-Power Diffie-Hellman assumption, with H modeled as a random oracle.*

Proof. The CI game adds bilateral structure with both receiver and sender identities.

Game 0. Setup and registration identical to RBEKS. In queries, $\mathcal{A}$ accesses ciphertext oracle $O_C(u_r, w)$ and trapdoor oracle $O_T(u_s, w)$ with appropriate restrictions. Challenge ciphertext for $(u^*_r, u^*_s, w^*_\theta)$:

$$ct_{0,1} = e(C_{k^*_r}, h_{n+1-u^{*\prime}_r}),$$

$$ct_{0,2} = \{e(\Lambda_{u^*_s,i}, g) \cdot e(h^{x^*_s}_{u^{*\prime}_s}, h_{n+1-u^{*\prime}_s})\}_{i\in[l^*_s]},$$

$$ct_{1,1} = {C_{k^*_r}}^{a_1}, \quad ct_{1,2} = {h_{n+1-u^{*\prime}_s}}^{a_2 \cdot x^{*-1}_s},$$

$$ct_{2,i} = \text{(similar)}, \quad ct_3 = {h_{u^{*\prime}_s}}^{a \cdot t^*_{w,\theta}}.$$

Recall $\Lambda_{u,j} = \prod_{\alpha\in[j],\hat{u}_\alpha\in\hat{D}_u} g^{x_{\hat{u}_\alpha} \cdot z^{\hat{u}'_\alpha} \cdot z^{n-u'+1}}$ where $\hat{D}_u = D_u \setminus \{u\}$.

Expanding: $ct_{1,1} = g^{a_1 \cdot x^*_r \cdot z^{u^{*\prime}_r}} \cdot \prod_{\hat{u}\in\hat{D}^*_r} g^{a_1 \cdot x_{\hat{u}} \cdot z^{\hat{u}'}}$.

Game 1. Randomize challenge with $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p^*$:

$$ct_3 = g^{(s_1+s_2)\cdot t^*_{w,\theta}}.$$

Lemma 5.5.6. *If DDH holds, then* $|Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}| \leq Adv^{DDH}(\lambda)$.

Proof. DDH reduction embeds challenge in relationship between $h_{u_s^{*\prime}}$ and encryption randomness. Set $a_2 = a$ from DDH (g, g^a, g^b, T), $h_{u_s^{*\prime}} = g^b$. Use T to construct ct_3. When $T = g^{ab}$, get Game 0; when $T = g^r$, get Game 1 with s_1, s_2 appropriately set. □

By random oracle argument (identical to RBEKS), $Adv^{(1)}_{\mathcal{A}} = 0$. Thus:

$$Adv^{\text{CI}}_{\mathcal{A}}(\lambda) \leq Adv^{\text{DDH}}(\lambda) = negl(\lambda).$$

□

Theorem 5.5.7. *RBAEKS is TI secure under GGM, the bilinear Diffie-Hellman assumption, and the n-Power Diffie-Hellman assumption, with H modeled as a random oracle.*

Proof. TI game reverses CI: adversary distinguishes challenge trapdoor for $(u_r^*, u_s^*, w_\theta^*)$:

$$td_{0,1} = \{e(\Lambda_{u_r^*,i}, g) \cdot e(h_{u_r^{*\prime}}^{x_r^*}, h_{n+1-u_r^*})\}_i,$$

$$td_{0,2} = e(C_{k_s^*}, h_{n+1-u_s^{*\prime}}),$$

$$td_{1,1} = {h_{n+1-u_s^*}}'^{b\cdot x_r^{*-1}\cdot t^*_{w,\theta}},$$

$$td_{1,2} = g^{b\cdot t^*_{w,\theta}\cdot x_s^*\cdot z^{u_s^{*\prime}}} \cdot \prod_{\hat{u}\in\hat{D}_s^*} g^{b\cdot t^*_{w,\theta}\cdot x_{\hat{u}}\cdot z^{\hat{u}'}},$$

$$td_{2,1,i} = \prod_{\alpha\in[i],\hat{u}\in\hat{D}_r^*} g^{b\cdot t^*_{w,\theta}\cdot x_r^{*-1}\cdot x_{\hat{u}}\cdot z^{\hat{u}'}\cdot z^{n-u_r^{*\prime}+1}},$$

$$td_{2,2} = g^{b\cdot t^*_{w,\theta}}, \quad td_3 = {h_{n+1-u_s^*}}'^{b}.$$

Game 1. Randomize with $s_1, s_2, s_3 \xleftarrow{\$} \mathbb{Z}_p^*$:

$$td_{1,1} = g^{s_1 \cdot t^*_{w,\theta}}, \quad td_{1,2} = g^{s_2} \cdot \prod_{\hat{u} \in \hat{D}^*_s} g^{b \cdot t^*_{w,\theta} \cdot x_{\hat{u}} \cdot z^{\hat{u}'}},$$

$$td_{2,1,i} = \prod_{\alpha} g^{s_3 \cdot x_r^{*-1} \cdot x_{\hat{u}} \cdot z^{\hat{u}'}}.$$

Lemma 5.5.8. *If DDH holds, then* $|Adv^{(0)}_{\mathcal{A}} - Adv^{(1)}_{\mathcal{A}}| \leq 3 \cdot Adv^{DDH}(\lambda)$.

Proof. Three sequential DDH reductions randomize $td_{1,1}$, $td_{1,2}$ leading term, and $td_{2,1,i}$ respectively. Each embeds DDH challenge in different component relationships:

- For $td_{1,1}$: embed in $h^{b \cdot x_r^{*-1}}_{n+1-u_s^{*\prime}} \to g^{s_1}$.
- For $td_{1,2}$: embed in $g^{b \cdot t \cdot x_s^* \cdot z^{u_s^{*\prime}}} \to g^{s_2}$.
- For $td_{2,1,i}$: embed to remove t factor.

Each reduction costs $Adv^{\text{DDH}}(\lambda)$. □

In Game 1, $td_{2,2} = g^{bt^*_{w,\theta}}$ with fresh random b and $t^*_{w,\theta}$ from random oracle is uniformly random, making $Adv^{(1)}_{\mathcal{A}} = 0$. Thus:

$$Adv^{\text{TI}}_{\mathcal{A}}(\lambda) \leq 3 \cdot Adv^{\text{DDH}}(\lambda) = negl(\lambda).$$

□

Theorem 5.5.9. *RBAE^{2}KS is CI secure under GGM, the bilinear Diffie-Hellman assumption, and the n-Power Diffie-Hellman assumption, with H modeled as a random oracle.*

Proof. CI for RBAE^{2}KS extends RBAEKS to keyword sets and expressive queries via LSSS.

Game 0. Setup and registration identical. Queries access $O_C(u_r, S)$ for keyword set $S = \{w_1, \ldots, w_m\}$ and $O_T(u_s, \mathbb{Q})$ for policy $\mathbb{Q}$ (LSSS matrix $\mathbf{M}$ with labeling ρ). Query restrictions ensure no queried ciphertext satisfies either challenge policy.

Challenge for $S^*_\theta = \{w^*_{\theta,1}, \ldots, w^*_{\theta,m}\}$ with receiver u^*_r, sender u^*_s:

Per-keyword components ($\sigma \in [m]$, $t^*_{\sigma,\theta} = H(w^*_{\theta,\sigma})$):

$$ct_{1,1,\sigma} = C_{k^*_r}^{a_1 \cdot t^*_{\sigma,\theta}}, \quad ct_{1,2,\sigma} = h_{n+1-u^{*\prime}_s}^{a_2 \cdot t^*_{\sigma,\theta} \cdot x_s^{*-1}}, \quad ct_{2,i,\sigma} = \text{(similar)}.$$

Shared authentication:

$$ct_{3,1} = {h_{n+1-u^{*\prime}_s}}^{a}, \quad ct_{3,2} = {C_{k^*_s}}^{a \cdot x_s^{*-1}}, \quad ct_{3,3,i} = {\Lambda_{u^*_s,i}}^{a \cdot x_s^{*-1}}.$$

Single-Keyword Security.

Lemma 5.5.10. *If RBAEKS is CI-secure, then RBAE^{2}KS is CI-secure against single-keyword attacks.*

Proof. Per-keyword components have identical structure to RBAEKS ciphertext. Any adversary distinguishing single keywords in RBAE^{2}KS directly translates to RBAEKS CI adversary. Thus $Adv_{\text{single}} \leq Adv^{\text{RBAEKS-CI}} = negl(\lambda)$. □

Multi-Keyword Security.

Lemma 5.5.11. *If CDH holds in $\mathbb{G}_T$, then adversary cannot distinguish S^*_0 from S^*_1 by combining multi-keyword information.*

Proof. Adversary cannot use trapdoors (query restrictions). To distinguish via cryptanalysis, adversary would need to compute $e(h_1, h_n)^{a \cdot t^*_{\sigma,\theta}}$ from available components.

From ciphertext, adversary can extract:

$$V_1 = e(h_1, h_n)^{a_1 \cdot t^*_{\sigma,\theta} \cdot x^*_r} \quad \text{(from } ct_{1,1,\sigma}, ct_{2,1,\sigma}\text{)},$$

$$V_2 = e(h_1, h_n)^{a_2 \cdot t^*_{\sigma,\theta} \cdot x^{*-1}_s} \quad \text{(from } ct_{1,2,\sigma}, ct_{2,2,\sigma,i}\text{)},$$

$$V_3 = e(h_1, h_n)^{a} \quad \text{(from } ct_{3,*}\text{)}.$$

To obtain target $e(h_1, h_n)^{(a_1+a_2)t^*_{\sigma,\theta}}$ for distinguishing, adversary must:

1. Remove x^*_r from V_1: requires $e(h_1, h_n)^{a_1 t^*_{\sigma,\theta}}$ from $V_1 = e(h_1, h_n)^{a_1 t^*_{\sigma,\theta} x^*_r}$ without knowing x^*_r.

2. Remove x^{*-1}_s from V_2: requires $e(h_1, h_n)^{a_2 t^*_{\sigma,\theta}}$ from V_2 without knowing x^*_s.

3. Multiply results.

Step 1 is CDH: computing G^α from $G^{\alpha\beta}$ where $G = e(h_1, h_n)$ without knowing $\beta = x^*_r$. If adversary succeeds, construct CDH solver:

CDH Challenge: Receive $(G, G^\alpha, G^\beta) \in \mathbb{G}^3_T$, goal: compute $G^{\alpha\beta}$.

Embedding: Set $G^{a_1 t^*_{\sigma,\theta}} = G^\alpha$, $x^*_r = \beta$ implicitly. Generate challenge so $V_1 = G^{\alpha\beta}$. If adversary distinguishes, it computed $G^{a_1 t^*_{\sigma,\theta}}$ without x^*_r factor, providing CDH solution $G^{\alpha\beta}$.

Thus $Adv_{\text{multi}} \leq Adv^{\text{CDH}}_{\mathbb{G}_T} = negl(\lambda)$. □

Conclusion.

$$Adv^{\text{CI}}_{\mathcal{A}}(\lambda) \leq Adv^{\text{RBAEKS-CI}}(\lambda) + Adv^{\text{CDH}}_{\mathbb{G}_T}(\lambda) = negl(\lambda).$$

□

Theorem 5.5.12. *RBAE²KS is TI secure under GGM, the bilinear Diffie-Hellman assumption, and the n-Power Diffie-Hellman assumption, with H modeled as a random oracle.*

Proof. TI game: adversary distinguishes which policy $\mathbb{Q}^*_\theta$ is embedded in challenge trapdoor.

Game 0. Challenge trapdoor for policy $\mathbb{Q}^*_\theta$ (LSSS $\mathbf{M}^{(\theta)}$, labeling ρ_θ) with $\mathbb{Q}^*_0, \mathbb{Q}^*_1$ identically structured but different keywords:

Shared components (identical to RBAEKS), plus per-keyword ($\mu \in [\ell]$, $w^*_{\mu,\theta} = \rho_\theta(\mu)$, $t^*_{\mu,\theta} = H(w^*_{\mu,\theta})$):

$$td_{3,1,\mu} = h_{u_s^{*\prime}}{}^{\mathbf{M}^{(\theta)}_\mu \mathbf{v}^\top \cdot x_r^{*-1} + b \cdot t^*_{\mu,\theta}}, \quad td_{3,2} = h_{n+1-u_s^{*\prime}}{}^{b \cdot x_r^{*-1}},$$

$$td_{3,3} = g^{b \cdot x_r^{*-1}},$$

where $\mathbf{v} = (b, v_2, \ldots, v_d)^\top$ with $v_2, \ldots, v_d \xleftarrow{\$} \mathbb{Z}_p$.

Information-Theoretic Security. Only $td_{3,1,\mu}$ depends on challenge keywords. Component has two terms:

1. $\mathbf{M}^{(\theta)}_\mu \mathbf{v}^\top \cdot x_r^{*-1}$: LSSS share (same structure for both θ).
2. $b \cdot t^*_{\mu,\theta}$: keyword-dependent term.

Key Observation: Term 1 acts as one-time pad for term 2. Since:

- $\mathbf{v}$ has $d-1$ uniformly random components, making $\mathbf{M}^{(\theta)}_\mu \mathbf{v}^\top$ uniformly random (from adversary's view).
- Division by unknown x^*_r preserves randomness.
- Query restrictions prevent adversary from obtaining satisfying keyword sets.

LSSS Reconstruction Barrier: Adversary might attempt reconstruction. If subset $I \subseteq [\ell]$ satisfies $\mathbb{Q}^*_\theta$, reconstruction coefficients $\{\gamma_\mu\}_{\mu \in I}$ give $\sum_{\mu \in I} \mathbf{M}^{(\theta)}_\mu \gamma_\mu =$

$(1, 0, \ldots, 0)$, thus:

$$\prod_{\mu \in I} td_{3,1,\mu}^{\gamma_\mu} = h_{u_s^{*\prime}}^{b \cdot x_r^{*-1}(1 + x_r^* \sum_{\mu \in I} \gamma_\mu t_{\mu,\theta}^*)}.$$

However, adversary cannot:

- Obtain keywords $\{w_{\mu,\theta}^*\}_{\mu \in I}$ (query restriction).
- Cancel x_r^* factor (doesn't know secret key).

Formal Argument: For any keyword values $t_{\mu,0}^*, t_{\mu,1}^*$:

$$(td_{3,1,\mu}^{(0)} \mid \text{public}) \equiv_{\text{dist}} (td_{3,1,\mu}^{(1)} \mid \text{public})$$

because $\mathbf{M}_\mu^{(\theta)} \mathbf{v}^\top \cdot x_r^{*-1}$ is uniformly random for both θ, perfectly masking $b \cdot t_{\mu,\theta}^*$.

Lemma 5.5.13. $Adv_{\mathcal{A}}^{TI}(\lambda) = 0$ *for any adversary (even unbounded).*

Proof. Challenge trapdoor distributions are identical for $\theta = 0$ and $\theta = 1$. Thus $\Pr[\theta' = \theta] = 1/2$ and $Adv_{\mathcal{A}}^{\text{TI}} = 0$. □

Conclusion. RBAE^{2}KS achieves perfect TI security via information-theoretic hiding from LSSS randomness, secret key masking, and query restrictions:

$$Adv_{\mathcal{A}}^{\text{TI}}(\lambda) = 0.$$

□

5.6 Evaluations

In this section, we present both theoretical and experimental evaluations of our schemes. We first analyze the computational and storage complexities theoretically, then present our experimental evaluation, including implementation details, benchmarks on synthesized data, and performance results on a large-scale real-world dataset. Both evaluations demonstrate that our schemes are efficient, effective, and scalable.

5.6.1 Asymmetric Pairing Adaptation. To adapt our schemes from the symmetric pairing setting (Type-I) to an asymmetric pairing setting for Type-III pairing, we calculate $h_{1,i} = g_1^{z^i}$ and $h_{2,i} = g_2^{z^i}$, and generate user keys using only $h_{1,*}$. Therefore, all C_k and Λ are elements in $\mathbb{G}_1$. Subsequently, we use $h_{2,\cdot}$ and g_2 to compute corresponding components in ciphertexts and trapdoors, except for the components used in calculating $e(h_{1,u'}, h_{2,n+1-u'})$, such as $ct_{3,1}$ and $td_{3,1,*}$.

5.6.2 Theoretical Evaluation. Given system capacity N and M registered users, setting parameter $n = B = \sqrt{N}$ for RBAEKS and RBAE^{2}KS (and $n = B = \sqrt{2N}$ for RBEKS) optimizes the public information: both `crs` and `pp` contain $\mathcal{O}(\sqrt{N})$ group elements. For RBAE^{2}KS, we set the query structure size to $m_1 \times m_2$ and the keyword set size for ciphertext to m_3 for simplicity, indicating that the query structure contains the same number of keywords as the ciphertext.

Computational Complexity. Inheriting from RBE, our RBEKS, RBAEKS, and RBAE^{2}KS schemes achieve efficiency in registration, update, and the number of updates, all sublinear in N, as highlighted in Table 7. It highlights that our schemes achieve efficiency on both client and server sides, achieving sublinear complexities cmpared to the state-of-the-art scheme F-PAEKS.

Table 7. Computational overheads for initial operations. N denotes the total number of users.

Schemes	**Setup**		**KGen**				**Register**		**# of Update for one user**
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	pairing	$\mathbb{G}_1$	pairing	
	Exp	Exp	Exp	Exp	Exp		Mul		
RBE Glaeser et al. (2023a)	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$2\sqrt{N}$	-	-	-	$\sqrt{N}$	$\sqrt{N}-1$	$\sqrt{N}$
RBEKS 7	$2\sqrt{2N}-1$	$2\sqrt{2N}-1$	$2\sqrt{2N}$	-	-	-	$\sqrt{2N}$	$\sqrt{2N}-1$	$\sqrt{2N}$
RBAEKS 8	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$2\sqrt{N}$	-	-	-	$\sqrt{N}$	$\sqrt{N}-1$	$\sqrt{N}$
RBAE^{2}KS 9	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$2\sqrt{N}$	-	-	-	$\sqrt{N}$	$\sqrt{N}-1$	$\sqrt{N}$
F-PAEKS L. Meng et al. (2024b)	**Setup**		**KGen**				**# of Pub Keys distributed to per user**		
	-	-	-	3	1	1	N		

Specifically, RBAEKS and RBAE^{2}KS achieve the same number of operations in system setup, key generation, and user registration. RBEKS demonstrates larger

overhead, approximately $\sqrt{2}$ times that of the former two schemes. Moreover, all schemes achieve $\sqrt{N}$ updates per user.

Table 8 shows that: (1) the complexity of our RBEKS scheme's encryption operations is $\mathcal{O}(\sqrt{N})$ while others are $\mathcal{O}(1)$; (2) the complexity of RBAEKS's test operation is $\mathcal{O}(1)$ while others are $\mathcal{O}(\sqrt{N})$; (3) the complexities of RBAE^{2}KS's encryption, trapdoor generation, and test operations are $\mathcal{O}(m_3\sqrt{N})$, $\mathcal{O}(m_1 + \sqrt{N})$, and $\mathcal{O}(1)$, respectively. The complexities of RBAE^{2}KS's encryption and trapdoor generation are sublinear compared to those of the state-of-the-art scheme F-PAEKS. The test operation complexities in RBEKS, RBAEKS, and RBAE^{2}KS are comparable to the state-of-the-art scheme.

Storage Complexity. Benefiting from the RBE framework, our schemes also achieve compactness of `crs`, `pp`, `hpk`, and `aux`, i.e., sublinear sizes in N, as shown in Table 9. It is easy to observe that all schemes achieve compactness of `crs`, `pp`, `hpk`, and `aux` with $\mathcal{O}(\sqrt{N})$ complexity. Specifically, RBAEKS and RBAE^{2}KS achieve the same storage overhead for public parameters and keys as RBE, while RBEKS achieves $\sqrt{2}$ times the complexity.

For ciphertext and trapdoor, RBEKS and RBAEKS achieve optimal storage overhead, requiring only a few group elements. Additionally, our RBAE^{2}KS scheme provides storage overhead for ciphertext and trapdoor linear in the number of involved keywords m_1 and m_3, which is optimal and comparable to the state-of-the-art expressive PAEKS scheme F-PAEKS introduced by L. Meng et al. (2024b).

System Scalability. Inheriting properties from Glaeser et al. (2023a)'s work, the system capacity is determined by parameter N chosen during setup. However, the system is not strictly bounded: if the number of registered users exceeds N, the system can

Table 8. Computational overheads for search operations. (Fully separated Ops)

Part I: Encryption (Enc)							
Schemes	$\mathbb{G}_1$ **Ops**		$\mathbb{G}_2$ **Ops**		$\mathbb{G}_T$ **Ops**		**Pairing**
	Exp	**Mul**	**Exp**	**Mul**	**Exp**	**Mul**	**P**
RBE Glaeser et al. (2023a)	1	-	1	-	1	1	1
RBEKS	3	-	2	-	-	-	1
RBAEKS	$2+\sqrt{N}$	-	2	-	-	-	1
RBAE^{2}KS	$m_3+(m_3+1)\sqrt{N}+1$	-	$2m_3+1$	-	-	-	1
F-PAEKS L. Meng et al. (2024b)	m_3	-	2	-	1	-	-
Part II: Trapdoor Generation (TGen)							
Schemes	$\mathbb{G}_1$ **Ops**		$\mathbb{G}_2$ **Ops**		$\mathbb{G}_T$ **Ops**		**Pairing**
	Exp	**Mul**	**Exp**	**Mul**	**Exp**	**Mul**	**P**
RBE Glaeser et al. (2023a)	-	-	-	-	-	-	-
RBEKS	$2\sqrt{N}$	-	3	-	-	-	-
RBAEKS	$1+\sqrt{N}$	-	3	-	-	-	1
RBAE^{2}KS	$m_1+\sqrt{N}+1$	-	4	-	-	-	1
F-PAEKS L. Meng et al. (2024b)	$3m_1$	$2m_1$	1	-	-	-	-
Part III: Test / Decryption							
Schemes	$\mathbb{G}_1$ **Ops**		$\mathbb{G}_2$ **Ops**		$\mathbb{G}_T$ **Ops**		**Pairing**
	Exp	**Mul**	**Exp**	**Mul**	**Exp**	**Mul**	**P**
RBE Glaeser et al. (2023a)	1	-	1	-	-	2	2
RBEKS	-	-	-	-	-	3	5
RBAEKS	-	-	-	-	-	3	5
RBAE^{2}KS	$3m_4$	$3m_4$	$2m_4$	$2m_4$	-	5	7
F-PAEKS L. Meng et al. (2024b)	-	$3m_4$	$5m_4$	$5m_4$	-	2	3

Table 9. Storage and communication overhead.

Schemes	crs		pp	
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$
RBE Glaeser et al. (2023a)	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$\sqrt{N}$	-
RBEKS	$2\sqrt{2N}-1$	$2\sqrt{2N}-1$	$\sqrt{2N}$	-
RBAEKS	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$\sqrt{N}$	-
RBAE^{2}KS	$2\sqrt{N}-1$	$2\sqrt{N}-1$	$\sqrt{N}$	-
F-PAEKS L. Meng et al. (2024b)	-	-	1	1
Schemes	aux	hpk	pk	
	$\mathbb{G}_1$	$\mathbb{G}_1$	$\mathbb{G}_1$	$\mathbb{G}_2$
RBE Glaeser et al. (2023a)	$\sqrt{N}-1$	$\sqrt{N}-1$	$\sqrt{N}-1$	-
RBEKS	$\sqrt{2N}-1$	$\sqrt{2N}-1$	$\sqrt{2N}-1$	-
RBAEKS	$\sqrt{N}-1$	$\sqrt{N}-1$	$\sqrt{N}-1$	-
RBAE^{2}KS	$\sqrt{N}-1$	$\sqrt{N}-1$	$\sqrt{N}-1$	-
F-PAEKS L. Meng et al. (2024b)	-	-	-	3
Schemes	ct			
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	
RBE Glaeser et al. (2023a)	1	1	1	
RBEKS	3	2	2	
RBAEKS	$\sqrt{N}+2$	2	$\sqrt{N}+1$	
RBAE^{2}KS	$m_3+(m_3+1)\sqrt{N}+1$	$2m_3+1$	$\sqrt{N}+1$	
F-PAEKS L. Meng et al. (2024b)	m_3	2	1	
Schemes	td			
	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_T$	
RBE Glaeser et al. (2023a)	-	-	-	
RBEKS	$2\sqrt{N}$	3	$2\sqrt{N}$	
RBAEKS	$\sqrt{N}+1$	3	$\sqrt{N}+1$	
RBAE^{2}KS	$m_1+\sqrt{N}+1$	4	$\sqrt{N}+1$	
F-PAEKS L. Meng et al. (2024b)	$2m_1$	1	-	

simply append N components to $\texttt{aux}$ to increase capacity to $2N$. This extension requires no system re-setup.

5.6.3 Experimental Evaluation. We conduct experiments on two datasets. The first experiment provides benchmarks with synthesized data for the proposed RBEKS, RBAEKS, and RBAE^{2}KS schemes. The second experiment demonstrates practicality with a large-scale real-world dataset for our primary contribution, the proposed RBAE^{2}KS scheme.

We implement our schemes in Python 3.13.7 on WSL with Ubuntu 24.04.3 LTS using the Charm 0.5 framework introduced by Akinyele et al. (2013) and the MNT224 curve for pairings, as it provides the best Type-III pairing performance in PBC, the default pairing library in Charm. All execution times are measured on a PC equipped with a 3.61 GHz Intel(R) Core(TM) i7-12700K 12-Core Processor and 32 GB RAM.

Keywords embedded in both ciphertexts and trapdoors follow the format "name:value". To construct a query structure, we first generate a Boolean formula using keywords combined with logical operators (AND, OR) and parentheses, then convert this formula to a Monotone Span Program (MSP) using the Lewko-Waters algorithm introduced by A. B. Lewko and Waters (2011). This conversion ensures that the matrix $\mathbf{M}$ contains only entries in $\{0, 1, -1\}$ and that the reconstruction coefficients γ are restricted to $\{0, 1\}$, thereby reducing the number of required exponentiations.

5.6.3.1 Benchmarks. We present benchmarks for our proposed RBEKS, RBAEKS, and RBAE^{2}KS schemes in this section.

For **Setup**, **KGen**, and **Register**, we test with user counts $N \in \{10^4, 10^5, 10^6, 10^7\}$. The results are shown in Table 10. We omit update operations since they simply retrieve the corresponding part of $\texttt{pp}$ as the helper key directly. For testing, we set the number of registered users to 100.

For other operations, we first synthesize a keyword space by selecting 100 English words as keyword names and another 100 English words as keyword values, yielding a total keyword space $\mathcal{K}$ of $100 \times 100 = 10{,}000$ keywords. For the RBAE^{2}KS scheme, we test encryption with $\{10, 20, \ldots, 100\}$ keywords. Meanwhile, we choose $\{5, 10, 15, \ldots, 50\}$ keywords combined with AND and OR operators to form keyword policies. The results are shown in Tables 11 and 12.

Results of Initial Operations. Table 10 presents the execution times for setup, user key generation, user registration, and update operations. Since the proposed RBEKS, RBAEKS, and RBAE^{2}KS schemes are built on the same RBE scheme, their execution times for these operations are very similar. Specifically, execution times for RBAEKS and RBAE^{2}KS are nearly identical, and RBEKS doubles the execution time for user key generation and registration while achieving approximately $1.4\times$ the execution time of RBAEKS and RBAE^{2}KS for setup, which approximates $\sqrt{2}$. All initial operations are efficient: even with 10^7 users, user registration in RBAEKS and RBAE^{2}KS takes no more than 10 seconds while system setup requires no more than half a minute. This is highly practical.

Table 10. Performance benchmarks for initial operations (in seconds)

N	Scheme	Setup	KGen	Register
10^4	RBEKS	1.27	0.119	0.735
	RBAEKS	0.896	0.0421	0.261
	RBAE^{2}KS	0.835	0.0422	0.264
10^5	RBEKS	3.67	0.375	2.33
	RBAEKS	2.59	0.133	0.826
	RBAE^{2}KS	2.61	0.134	0.832
10^6	RBEKS	12.1	1.21	7.44
	RBAEKS	8.21	0.422	2.61
	RBAE^{2}KS	8.56	0.421	2.62
10^7	RBEKS	43.9	3.97	23.8
	RBAEKS	28.9	1.36	8.32
	RBAE^{2}KS	29.3	1.34	8.31

Results of Search Operations. Table 11 presents the execution times for encryption, trapdoor generation, and test operations for RBEKS and RBAEKS. Execution times for RBEKS and RBAEKS are highly similar since they perform similar calculations (as demonstrated in Sections 5.4.2 and 5.4.1). To encrypt one keyword and test a ciphertext-trapdoor pair, RBEKS and RBAEKS achieve execution times of approximately 13 ms. For trapdoor generation to query a keyword, their execution times are approximately 9.2 ms. These results demonstrate our schemes' efficiency on both client and server sides.

Table 11. Performance benchmarks of the RBEKS scheme and the RBAEKS scheme for search operations with 10^7 capacity (in seconds)

Schemes	**Enc**	**TGen**	**Test**
RBEKS	0.0132	0.0092	0.0137
RBAEKS	0.0126	0.0092	0.0136

Table 12 shows the benchmarks for the RBAE^{2}KS scheme's encryption, trapdoor generation, and test operations with varying keyword counts. Notably, the test operation is performed with various trapdoor sizes and a 100-keyword ciphertext. In the worst case, it takes only 0.8 seconds to generate a ciphertext with 100 keywords and no more than 40 milliseconds to generate a trapdoor with 50 keywords. The test algorithm achieves effective performance with execution times under 40 milliseconds. This demonstrates that RBAE^{2}KS is significantly efficient.

5.6.3.2 Experiments on Large-Scale Real-World Dataset.

Keyword Extraction. We use the Enron email corpus Klimt and Yang (2004), a widely used searchable encryption benchmark. From the "sent_items" category (37,920 emails, 8,951 addresses), we generate 64,747 sender-receiver pair encryptions. Keywords are extracted by removing stop words and stemming with **nltk**.

Table 12. Performance benchmarks of the RBAE^{2}KS scheme for search operations with 10^7 capacity (in seconds)

# of kw in `ct`	**Enc**	# of kw in `td`	**TGen**	**Test**
10	0.0882	5	0.0185	0.0194
20	0.1696	10	0.0203	0.0198
30	0.2546	15	0.0230	0.0207
40	0.3356	20	0.0250	0.0222
50	0.4153	25	0.0272	0.0232
60	0.5047	30	0.0306	0.0268
70	0.5877	35	0.0323	0.0275
80	0.6661	40	0.0351	0.0312
90	0.7582	45	0.0373	0.0342
100	0.8369	50	0.0387	0.0378

We generate keyword names and values as follows: (1) rank keywords by frequency per email; (2) vectorize keywords using **FastText** implemented by Joulin, Grave, Bojanowski, and Mikolov (2016); (3) cluster into 100 groups using **k-means** introduced by Lloyd (1982); (4) construct keyword names as `cluster-X-rank-Y` where X is the cluster ID and Y is the frequency rank. For example, "computer" with the second highest frequency in cluster 5 becomes `(name: ''cluster-5-rank-2'', value: ''computer'')`. This enables relevance-based queries while reducing duplicate matching and preserving value privacy.

Each email contains an average of 74 keywords, with ¿75% containing ¡100 keywords and ¿90% containing ¡200 keywords. We set $N = 10^7$ and test search operations over all 64,747 encrypted emails using trapdoors with $\{5, 10, 15, \ldots, 50\}$ randomly selected keywords.

Results. Table 13 shows search times for various query sizes. Our RBAE^{2}KS achieves efficient performance: 5-keyword queries complete in under 1 minute, while 50-keyword queries complete in approximately 9 minutes, demonstrating practical efficiency for large-scale deployments.

Table 13. Search performance for various-sized keyword policies on the encrypted dataset (in seconds)

# of kw in `td`	Search time
5	52.13
10	107.32
15	155.63
20	206.87
25	260.48
30	297.11
35	355.92
40	420.73
45	455.45
50	524.78

5.7 Conclusion

In this chapter, we have proposed a registration-based authenticated encryption with expressive keyword search (RBAE^{2}KS) scheme and applied similar techniques to develop two foundational single-keyword search schemes: registration-based encryption with keyword search (RBEKS) and registration-based authenticated encryption with keyword search (RBAEKS). These three schemes achieve the highest efficiency level among their respective primitives and are comparable to state-of-the-art RBE schemes and the expressive asymmetric searchable encryption scheme FEASE. The absence of user revocation remains a limitation of our schemes. Future work will address this shortcoming and further advance research in this domain.

CHAPTER VI

PEKS IN THE HYBRID TRUST SETTING

This chapter is based on the paper *"M-EDESE: Multi-Domain, Easily Deployable, and Efficiently Searchable Encryption,"* published in the *Proceedings of the 17th International Conference on Information Security Practice and Experience (ISPEC 2022)*. I was the primary author, responsible for the scheme design, security analysis, implementation, and manuscript preparation. Dr. Yingjiu Li provided conceptual guidance and editorial revisions. Jianting Ning and Robert H. Deng contributed to the refinement of the manuscript.

The protocols developed in Chapters IV and V establish strong theoretical foundations for secure and trust-minimized search. However, as identified in our earlier analysis, a significant gap remains between cryptographic design and real-world system deployability, particularly in multi-domain environments like healthcare networks. This chapter bridges that gap by presenting M-EDESE, a system architecture that prioritizes 'easy-deployability' and cross-domain interoperability without requiring additional cooperation from cloud storage providers.

6.1 Introduction

Secure searching service is a key component of secure cloud data services and cloud-based apps, such as Gmail, Facebook, and Outlook. Many searchable encryption schemes have been proposed to provide such secure searching service (e.g., Amanatidis, Boldyreva, and O'Neill (2007); Boneh et al. (2004a); Boneh, Sahai, and Waters (2012); Boneh and Waters (2007); Bösch, Hartel, Jonker, and Peter (2014); Cash et al. (2014); Chai and Gong (2012); Chang and Mitzenmacher (2005); Curtmola, Garay, Kamara, and Ostrovsky (2011); W. He et al. (2014); Lau et al. (2014)). One practical scheme, named

easily deployable and efficiently searchable encryption (EDESE), was proposed by Lau et al. (2014). EDESE is widely compatible and highly efficient for ease of deployment.

Rigerous research has been conducted on EDESE (e.g., Boneh et al. (2004a, 2012); Bösch et al. (2014); W. He et al. (2014); Poh, Chin, Yau, Choo, and Mohamad (2017); Yin et al. (2019)) following the initial work Song et al. (2000). The initial EDESE was designed for single-user settings. To extend EDESE to multi-user settings, PEKS was proposed by Boneh et al. (2012), which is the first public-key-based searchable encryption satisfying EDESE requirements. Then, MuED was introduced by Bao et al. (2008), which improves query generation performance significantly in multi-user settings. Recently, CP-ABSE was proposed by Yin et al. (2019), which increases the scalability of EDESE in multi-user settings based on attribute-based searchable encryption. Most of the existing EDESE schemes, regardless of being designed in single-user settings or multi-user settings, work in single autonomous domains, in which all users are managed by a single authority. In practice, however, industrials need a secure searching service in multi-domain settings so that users in one autonomous domain can search for useful information from collaborative domains (e.g., in supply chain management).

To fill this gap, we propose a multiple domain searchable encryption based on EDESE, called Multi-Domain, Easily Deployable, Efficiently Searchable Encryption (M-EDESE). M-EDESE leverages the easy deployment of EDESE to allow users to query encrypted keywords among multiple domains. In particular, M-EDESE achieves the following advantages.

– **Secure Searching across Multiple Domains.** M-EDESE enables secure keyword searching across multiple domains. It allows any single domain to be enrolled at any time without a re-initialization of the whole system. Any domain can establish a unilateral searching collaboration relationship with other domains. In this

unilateral relationship, the host domain provides a searching service to the partner domains such that any authorized user under a partner domain can query encrypted keywords from the host domain. Each domain in M-EDESE is fully autonomous in managing its own users and deciding on its own partners.

– **Easy deployment.** Easy deployment is an essential achievement of EDESE as required in many EDESE applications such as ShadowCrypt implemented by W. He et al. (2014) and Mimesis Aegis implemented by Lau et al. (2014). M-EDESE inherits the easy deployment property of EDESE in the multiple domains setting. Any domain in M-EDESE can work directly with any cloud storage without making any changes to the keyword searching services provided by the cloud storage.

Besides the above achievements, M-EDESE offers secure searchable encryption with **query privacy, query unforgeability,** and **revocability**.

– **Query Privacy.** Query privacy introduced by Curtmola et al. (2011) is a foundational secure requirement for all searchable encryption. It requires that no server providing searching services can determine the underlying keyword of any query issued by a user following searchable encryption, even though the server can observe all access patterns of users' queries to the data in its storage. Furthermore, it requires that no sensitive information about users' queries can be derived by the server from its observed information.

– **Query Unforgeability.** In M-EDESE, queries are generated by a user's secret key, which is distinct from others' keys. It is a basic requirement that neither user nor server can generate a legitimate query on behalf of a user without the corresponding secret key like Bao et al. (2008). In the multi-domain setting, M-EDESE should

also guarantee that neither the user nor the server can generate a legitimate query from any domain without the corresponding secret key of the domain.

– **Revocability.** M-EDESE provides effective user revocation and partner revocation. User revocation allows a domain the capacity to manage its users while partner revocation provides a host domain to manage its partner domains. A revoked user is not allowed to access the searching services provided by their host domain or any of its partner domains. If a partner domain is revoked by a host domain, then all users belonging to the revoked partner domain can no longer access the searching services provided by the corresponding host domain.

Chapter Organization. In this chapter, we demonstrate a detailed construction of M-EDESE (section 6.3) and present the formal proofs regarding query privacy, query unforgeability, and revocability (section 6.4). Then, we provide our evaluations on M-EDESE's performance (section 6.5) and conclusion (section 6.6).

6.2 Preliminaries

M-EDESE is constructed based on two preliminaries, including bilinear maps and BLS short signature.

6.2.1 Bilinear Maps.

Let $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ be three cyclic multiplicative groups of prime order p. A bilinear map is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and is said to be an admissible bilinear map if the following properties hold.

1. Bilinearity: for all $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2. Non-degeneration: if g_1 is a generator of $\mathbb{G}_1$ and g_2 is a generator of $\mathbb{G}_2$, then $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.
3. Computability: there exists an efficient algorithm to compute $e(g_1, g_2)$ for any $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$.

We say that $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are bilinear map groups if there exists a bilinear pairing function $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.

6.2.2 BLS Short Signature. Boneh, Lynn, and Shacham (2001) proposed a short signature scheme based on the bilinear map. The BLS short signature consists of three functions: keygen, sign, and verify. We recall the brief definition of the BLS short signature as follows: Let $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be defined as the above (section 6.2.1), g_1 be a generator of $\mathbb{G}_1$, $h : \{0,1\}^* \rightarrow \mathbb{G}_2$ be a collision-resistant hash function. A user's key pair is generated by the keygen algorithm, which runs as $(x, y) \leftarrow (x \in \mathbb{Z}_p^*, y = g_1^x)$. Then, the signature on a message m is defined as $\sigma = h(m)^x$ that is generated by the signing algorithm using a user secret key x. The signature verification is to check $e(g_1, \sigma) \stackrel{?}{=} e(y, h(m))$. The BLS short signature implements existential unforgeability if h is modeled as a random oracle, which means the adversary can not forge an eligible signature on behalf of a target user without its secret key.

6.3 Construction of M-EDESE

We will use the notations as shown in Table 1 in the construction of M-EDESE.

6.3.1 System Architecture. M-EDESE System mainly consists of seven entities: *Host Domain (HD), Partner Domain (PD), Data User (DU), Data Owner (DO), Key Generation Center (KGC), Operational Center (OC),* and *Cloud Storage Provider (CSP)*. These entities are defined below.

- **Host Domain (*HD*) and Partner Domain (*PD*).** Host Domain provides a searching service for its Partner Domain after establishing a collaboration relationship. PD needs to apply to the HD before building a relationship. Both PD and HD can revoke their users. In addition, HD can also revoke its PDs.

- **Key Generation Center (*KGC*).** Each domain maintains a key generation center (KGC). KGC only accounts for its domain and generates all keys for the entities in

Table 14. Notations

Symbol	Descriptions
GP	the global parameters generated by the coordinator domain's KGC
uid	user identity
did_p	partner domain's identity
did_h	host domain's identity
MK_{did}	a master key corresponding to the domain did
QK_{uid}	a query key corresponding to the User uid
SK_{uid}	a search key corresponding to the User uid
$DT_{did_p}^{did_h}$	a delegation key generated by a partner domain did_p and sent to a host domain did_h
$PSK_{did_p}^{did_h}$	a domain search key corresponding to the partner domain did_h for the domain did_p
$PQK_{did_p}^{did_h}$	a domain query key corresponding to the partner domain did_h for the domain did_p
I_{kw}	an index associated with the keyword kw
$I_{kw,did}$	an index associated with the keyword kw owned by the domain did
$USKL$	a list of user search keys
$PQKL$	a list of domain query keys
$PSKL$	a list of domain search keys

the domain. Moreover, KGCs of PD and HD interact with each other to agree on the establishment of a collaboration relationship.

- **Data Owner (*DO*) and Data User (*DU*).** Data Owner encrypts keywords into query tokens and uploads them with associated encrypted files to the cloud storage via the domain's operational center which is mentioned later. Data User can generate query tokens to query keywords from its HD's data and the corresponding PDs' data on the cloud storage.

- **Operational Center (*OC*).** Each domain maintains an operational center (OC) online. In its operations, an operational center receives user query tokens and domain query tokens and transfers them into indexes. Besides, OC can revoke its HD users and PDs.

- **Cloud Storage Provider (CSP).** Cloud Storage provider (CSP) stores encrypted files and associated encrypted keywords (named indexes in M-EDESE) for any domain and provide keyword searching services based on exact match (which is the same as the search services provided by existing cloud service providers on plaintext data). The storage and searching services provided by CSP can be accessed by all entities. Apart from all indexes and encrypted files, the global public parameters are stored on CSP with integrity protection.

6.3.2 Algorithm. The M-EDESE system involves 6 algorithms, consisting of initial setup, user enrollment, collaboration establishment, data uploading, search, and revocation. Below are their definitions.

Initial Setup. This process setups the entire base environment, which includes keys of involved domains and a set of global parameters GP. The initial setup consists of a global setup phase and a domain setup phase.

– **Global Setup Phase:**

1. The coordinator, one of KGCs, chooses a bilinear map $e : (G)_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and let g_1, g_2 be the generators of $\mathbb{G}_1, \mathbb{G}_2$, respectively.
2. It chooses a collision-resistant hash function: $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{G}_2$.
3. The coordinator publishes the global public parameters GP to CSP with integrity protection, where $GP = \{g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathcal{H}\}$.
4. Since GP is public shared on CSP, we assume all entities already obtain it before any operation.

– **Domain Setup Phase:** Each domain did performs the following operations:

1. Its KGC randomly chooses $x_{did} \in_R \mathbb{Z}_p^*$ as this domain's master key MK_{did} and stores it in the KGC's local storage.
2. Meanwhile, the domain's OC initializes 3 lists $USKL, PSKL$, and $PQKL$ to maintain its users' search keys, the PDs' search keys, and the PDs' query keys.

User Enrollment. When a user uid registers in the M-EDESE system under the domain did, the KGC of the domain did firstly randomly chooses $y_{uid} \in_R \mathbb{Z}_p^*$ as the user's query key QK_{uid}. Then, the KGC computes $k_{uid} = g_1^{\frac{x_{did}}{y_{uid}}}$ as the user's search key SK_{uid}, where x_did is the domain's master secret key. After that, the KGC uses a secure communication channel to send SK_{uid} to the domain's OC and send QK_{uid} to the user uid. Once the OC receives SK_{uid}, it updates its list $USKL$ with the pair of (uid, SK_{uid}).

Data Uploading. The data uploading process allows DU to upload an encrypted file and a set of associated indexes to the encrypted keywords for the file into CSP. When a DU uid uploads to CSP an encrypted file E (e.g., encrypted by AES) and a set of

the indexes cooresponding to a set S_{kw} of n keywords $kw_1, kw_2, \cdots, kw_n$ which are associated with E, DU performs the following operations.

1. The DU computes $S_q = \{q_i : q_i = \mathcal{H}(kw_i)^{y_{uid}}\}_{i\in[1,n]}$.
2. Then, the DU sends the tuple (uid, E, S_q) to the OC of the domain to which the DU belongs.
3. After receiving (uid, E, S_q), the OC firstly retrieves the DU's search key SK_{uid} from the list $USKL$ by uid.
4. Secondly, the OC computes $S_I = \{I_i : I_i = e(k_{uid}, q_i)\}_{i\in[1,n]}$.
5. Finally, the OC uploads the pair of (E, S_I) to the CSP.

Collaboration Estabilishment. When a domain did_p or its users need to access the searching service from another domain did_h, the domain did_p must establish a collaboration relationship with domain did_h as PD PD while domain did_h acts as HD. Assuming that the involving domains can verify each other's signatures using a signature system (e.g., PKI), they establish a collaboration relationship through the following steps.

1. At first, PD's KGC randomly chooses $s_{did_p}^{did_h} \in_R \mathbb{Z}_p^*$ as the PD's query key $PQK_{did_p}^{did_h}$ and computes $d_{did_p}^{did_h} = \frac{1}{x_{did_p} s_{did_p}^{did_h}}, d'^{did_h}_{did_p} = g_1^{s_{did_p}^{did_h}}$, where x_{did_p} is PD's master secret key.
2. Secondly, PD's KGC produces a digital signature σ_p on $DT_{did_p}^{did_h}$, where $DT_{did_p}^{did_h} = (d_{did_p}^{did_h}, d'^{did_h}_{did_p})$.
3. PD's KGC sends the pair of $(\sigma_p, DT_{did_p}^{did_h})$ to HD's KGC.
4. After receiving $(\sigma_p, DT_{did_p}^{did_h})$, HD verifies the signature σ_p firstly. If it is valid, it proceeds to the next step; otherwise, does nothing.

5. HD's KGC computes $z_{did_p}^{did_h} = x_{did_h} d_{did_p}^{did_h}, z'^{did_h}_{did_p} = d'^{did_h}_{did_p}$.

6. Then, HD's KGC sends PD's search key $PSK_{did_p}^{did_h}$ to HD's OC, and HD's OC stores it into $PSKL$, where $PSK_{did_p}^{did_h} = \{z_{did_p}^{did_h}, z'^{did_h}_{did_p}\}$.

7. HD's KGC also produces a digital signature σ_h on $DT_{did_p}^{did_h}$ and sends to PD's KGC.

8. PD's KGC verifies the signature σ_p. If it is valid, it proceeds to the next step; otherwise, does nothing.

9. Afterward, PD's KGC sends $PQK_{did_p}^{did_h}$ to PD's OC, and PD's OC stores it into $PQKL$.

Finally, regarding this collaboration relationship, HD's OC maintains PD's search key in $PSKL$ while PD's OC maintains its query key in $PQKL$.

Search. In M-EDESE, there are two search cases. One is search within a single domain, and the other is search across domains. The first processes of these two cases are the same: DU with uid produces a user query token q for a queried keyword kw.

- The DU computes the user query token as $q = \mathcal{H}(kw)^{y_{uid}}$, where y_{uid} is DU's query key.

Then, the subsequent processes are different. Search within a single domain only requires the domain's OC's cooperation, while search across domains needs both HD's and PD's OCs' operation.

- **Within Single Domain:**

  1. The DU sends uid and q to the OC of the domain which owns the DU.
  2. The OC obtains DU's search key k_{uid} from its $USKL$ and generates index $I_{kw} = e(k_{uid}, q)$.

3. Then, the OC queries I_{kw} from the corresponding CSP to retrieve the list of encrypted files associated with it.
4. Finally, the OC returns the retrieved list of encrypted files to the DU.

– **Cross Domain:**

1. Besides uid and q, the DU sends the id did_h of the queried HD to the OC of the PD did_p which owns the DU.
2. The OC obtains DU's search key k_{uid} from its $USKL$ by uid and obtains the PD's query key $PQK_{did_p}^{did_h}$ from $PQKL$ by did_h.
3. Next, the OC computes $I_{kw,did_p} = e(k_{uid}, q), r_{did_p}^{did_h} = I_{kw,did_p}^{s_{did_p}^{did_h}}, \sigma_r = \mathcal{H}(r_{did_p}^{did_h})^{s_{did_p}^{did_h}}$, and $rq = (r_{did_p}^{did_h}, \sigma_r)$, and sends rq and did_p to the OC of the queried HD.
4. After receiving rq and did_p, HD's OC obtains the PD's search key $PSK_{did_p}^{did_h}$ first from its $PSKL$.
5. Then, HD's OC verifies $e(g_1, \sigma_r) \stackrel{?}{=} e(z'^{did_h}_{did_p}, \mathcal{H}(r_{did_p}^{did_h}))$. If they are identical, go to the next step; otherwise, do nothing.
6. HD's OC computes $I_{kw,did_h} = (r_{did_p}^{did_h})^{z_{did_p}^{did_h}}$ and queries it from the CSP to retrieve a list of encrypted files.
7. Finally, HD's OC sends the retrieved list to the DU via the PD's OC.

Revocation.. A domain's OC in M-EDESE can revoke any of the domain's users and partner domains.

– **User Revocation:** The OC revokes a user by deleting the search key from $USKL$ indexed by uid. As the OC no longer possesses the user's search key, the user can not query the OC's domain anymore.

– **Collaboration Revocation:** HD's OC revokes PD's search key from $PSKL$ that is indexed by did_p and informs the PD. Then, PD's OC deletes its search key regarding the HD from its $PQKL$ indexed by did_h. Since HD's OC no longer stores PD's search key, the process of searching across domains is blocked.

6.4 Security Analysis

In this section, we prove that M-EDESE satisfies the requirements for query privacy, query unforgeability, and revocability.

6.4.1 Query Privacy. We will use notations defined below to prove that M-EDESE satisfies the requirement of query privacy.

Given an encrypted file e_i, we use $id(e_i)$ to denote the identifying formation that is uniquely associated with e_i, such as its database position or its memory location.

Given a user query token q_{kw} sent by the user uid with the queried domain did_h and its reply $rs(q_{kw})$, which is the output of the search process taking q_{kw} as input and is a set of encrypted files associated with q_{kw}, we define $\Omega(q_{kw})$ be a tuple of $(uid, did_h, id(rs(q_{kw})))$, where $id(rs(q_{kw}))$ represents the identifying information of each encrypted file in $rs(q_{kw})$, let $Q_t = \{q_1, \cdots, q_t\}$ be a sequence of t user query tokens sent from users to the OC in the domain did_h, and let $RS_t = \{rs(q_1), \cdots, rs(q_t)\}$ be the corresponding replies, where $t \in \mathbb{N}$ and is polynomial bounded.

Similarly, given a domain query token rq_{kw} sent by a querier domain did_p to a queried domain did_h and its reply $rs(rq_{kw})$, we define $\Omega(rq_{kw}) = (did_p, did_h, id(rs(rq_{kw})))$, where $id\,(rs(rq_{kw}))$ represents the identifying information of each entry in $rs(rq_{kw})$, let $RQ_l = \{rq_1, \cdots, rq_l\}$ as a sequence of l domain query tokens sent from OCs of the domain did_h's PDs to the domain did_h's OC, and $RS' = \{rs(rq_1), \cdots, rs(rq_l)\}$ be the corresponding replies, where $l \in \mathbb{N}$ is polynomial bounded.

An adversary is an honest but curious OC, and we name the domain to which the adversary OC belongs as the adversary domain. We define V as the view of an adversary over t user query tokens and l domain query tokens as the transcript of the interactions between the adversary and users/PDs of the adversary domain, together with some common knowledge. We have $V = (D, USKL, PSKL, PQKL, Q_t, RQ_l, RS_t, RS'_l)$, where $D = (I, ED), USKL = \{USK_1, USK_2, \cdots\}, PSKL = \{PSK_1, PSK_2, \cdots\}, PQKL = \{PQK_1, PQK_2, \cdots\}$, D is the encrypted database, I is the set of generated indexes based on keywords, ED is the set of encrypted files involving a ciphertext space $\mathbb{E}$.

Following the notation from Curtmola et al. (2011)'s work, the $trace$ of the t user query tokens and l domain qeury tokens is defined to be: $T = (|D|, \Omega(q_1), \cdots, \Omega(q_t), \Omega(rq_1), \cdots, \Omega(rq_l), |U|, |P|, |H|)$, which contains all the information that we allow a **simulator** (outsider observer who can only observe the communications between entities during search processes) to obtain. Note that $|D| = (|I|, |ED|)$, $|I|$ is the number of indexes based on keywords, $|ED|$ denotes the number of encrypted files, $|U|$ equals the number of entries in $USKL$, representing the number of the adversary domain's users, $|P|$ equals the number of entries in $PSKL$, which is the number of the adversary's PDs, and $|H|$ equals the number of entries in $PQKL$, which represent the number of the adversary's collaborating HDs.

Definition 13. *An M-EDESE achieves query privacy if for all database D, for all $t, l \in \mathbb{N}$, for all PPT algorithm $\mathcal{A}$, there exists a PPT algorithm (the **simulator**) $\mathcal{A}^*$, such that for all view V and trace T, for any function f:*

$$|Pr[\mathcal{A}(V) = f(D, KW_t, KW'_l)]$$
$$-Pr[\mathcal{A}^*(T) = f(D, KW_t, KW'_l)]| < v(\kappa),$$

where the probability is taken over the internal coins of $\mathcal{A}$ and $\mathcal{A}^*$.

The notion of query privacy requires that all information on the original database and the queried keywords that can be computed by the adversary from the transcript of interactions it obtains (i.e., V) can also be computed by the simulator from what it is allowed to know (i.e., T). In other words, a system satisfying query privacy does not leak any information beyond the information we allow the adversary to have.

6.4.1.1 Proof. To construct a *PPT* simulator $\mathcal{A}^*$ such that for all $t, l \in \mathbb{N}$, for all *PPT* algorithm $\mathcal{A}$, all functions f, given the trace T, $\mathcal{A}^*$ can simulate $\mathcal{A}(V)$ with non-negligible probability. More specifically, we need to show that $\mathcal{A}^*(T)$ can generate a view V^* which is computationally indistinguishable from V, the actual view of $\mathcal{A}$.

Recall that T and V definitions are as follows,

1. Given $t = 0$ and $l = 0$, which imply $Q = \emptyset, RQ = \emptyset, RS = \emptyset$, and $RS' = \emptyset$, $\mathcal{A}^*$ builds $V^* = (D^* = (I^*, ED^*), USKL^*, PSKL^*, PQKL^*)$ as follows.

   - To construct D^*, for $1 \leq i \leq |I|, 1 \leq j \leq |ED|$, it selects $I^*_{kw_i} \in_R \mathbb{G}^*_T$ and $E(m_j)^* \in_R \mathbb{E}$.
   - Then, it computes $USKL^* = \{k^*_{uid_i} : k^*_{uid_i} \in_R \mathbb{G}^*_1, uid_i \in_R \mathbb{N}\}_{i \in [1,|U|]}$, $PSKL^* = \{(z^*_{did_j}, z'^*_{did_j}) : z^*_{did_j} \in_R \mathbb{Z}^*_p, z'^*_{did_j} \in_R \mathbb{G}^*_1, did_j \in_R \mathbb{N}\}_{j \in [1,|P|]}$, and $PQKL^* = \{s^{*did'_k} : s^{*did'_k} \in \mathbb{Z}^*_p, did'_k \in \mathbb{N}\}_{k \in [1,|H|]}$, where did_j is a PD's identity of the $\mathcal{A}$ domain and did'_k is a collaborator domain's identity of the $\mathcal{A}$ domain.

   It is easy to check that V^* and V are computationally indistinguishable when $E(\cdot)$ is a pseudorandom permutation and $\mathcal{H}$ is a pseudorandom function, which is demonstrated below.

- If $E(\cdot)$ is a pseudorandom permutation, ED^* and ED are computationally indistinguishable, because a real $E(m)$ and a simulated $E(m)^* \in_R \mathbb{E}$ are both random values in the ciphertext space $\mathbb{E}$.
- If $\mathcal{H}$ is pseudorandom function, a real index $I_{kw} = e(g_1, \mathcal{H}(kw))^x$ and a simulated index I^*_{kw} are computationally indistinguishable due to they are both random elements in the group $\mathbb{G}^*_T$.
- A real user search key $k_{uid} = g_1^{\frac{x}{y_{uid}}}$ and a simulated user search key k^*_{uid} are computationally indistinguishable as both of them are random elements in the group $\mathbb{G}^*_1$.
- A real domain search key $(z_{did} = \frac{x}{x_{did} s_{(did)}}, z_{(did)} = g_1^{s_{(did)}})$ and a simulated partner search key (z^*_{did}, z'^*_{did}) are computationally indistinguishable as both of z are random elements in the group $\mathbb{Z}^*_p$ and both of z' are random elements in the group $\mathbb{G}^*_1$.
- A real domain query key s^{did} and a simulated domain query key s^{*did} are computationally indistinguishable as both of s are random elements in the group $\mathbb{Z}^*_p$.

2. Given $t > 0, l > 0$, and a function $select(i, N)$ which is defined to return the i-th entry in the set N, $\mathcal{A}^*$ builds $V^* = (D^* = (I^*, ED^*), USKL^*, PSKL^*, PQKL^*, Q^*_t, RQ^*_l, RS^*_t, RS'^*_l)$ as follows. Recall that $\Omega(rq) = (did_p, did_h, id(rs(rq)))$, $\mathcal{A}^*$ obtains $\mathcal{A}$ domain identity $did_{\mathcal{A}} = did_h$ from $\Omega(rq_1)$ contained in T. Next, $\mathcal{A}^*$ setups five sets $UID = \emptyset, DID = \emptyset, X^* = \{x^*_i\}_{i\in[1,|P|+|H|]}$, $Y^* = \{y^*_j\}_{j\in[1,|U|]}$, and $S^* = \{s^*_k\}_{k\in[1,|P|]}$, s.t. $x^*_i, y^*_j, s^*_k \in_R \mathbb{Z}^*_p$. Then, it computes the below equation, where x^* is regarded as the $\mathcal{A}$ domain's master secret key.

Then, it proceeds following operations.

$$x^* = \prod_{i=1}^{|P|+|H|} \prod_{j=1}^{|U|} \prod_{k=1}^{|P|} x_i^* y_j^* s_i^*$$

- To construct ED^*, $\mathcal{A}^*$ performs the same operations as the case of $t = 0$ and $l = 0$ to obtain the $ED^* = \{E(m_i)^*\}_{i\in[1,|ED|]}$.
- Next, it computes $PSKL^* = \{(z_i^*, z'^*_i) : x_i^* = select(i, X^*), s_i = select(i, S^*), z_i^* = \frac{x_i^*}{x^* s_i^*}, z'^*_i = g_1^{s_i^*}\}_{i\in[1,|P|]}$, $USKL^* = \{k_j^* : y_j^* select(j, Y^*), k_j^* = g_1^{\frac{x^*}{y_j^*}}\}_{j\in[1,|U|]}$, and $PQKL^* = \{s^{*did'_k} : did'_k \in_R \mathbb{N}, s^{*did'_k} \in_R \mathbb{Z}_p^*\}_{k\in[1,|H|]}$.
- To construct Q_t^*, recall that from $\Omega(q_1), \cdots, \Omega(q_t)$ contained in T, $\mathcal{A}^*$ can determine which user query tokens ask the same keyword and which user query tokens are issued by the same user. Hence, regarding to each $\Omega(q_i) = \{uid_i, did_i, id(rs(q_i))\}$ where i ranges from 1 to t, it computes q_i^* as follows.
  (a) $\mathcal{A}^*$ finds out $j \in [1, |UID|]$ such that $uid'_j = uid_i$, where $uid'_j \in UID$. If no such j value exists, it computes $UID = UID \cup \{uid_i\}$ and sets $j = |UID|$.
  (b) If there does not exist $1 \leq k < i$ such that $id(rs(q_k)) = id(rs(q_i))$, which means q_i and q_k ask the same keyword, $\mathcal{A}^*$ selects a random element $w_i \in \mathbb{G}_2^*$; otherwise, it sets $w_i = w_k$.
  (c) Then, $\mathcal{A}^*$ sets $y_j = select(j, Y^*)$ as the user uid_i's query key and calculates $q_i^* = w_i^{y_j^*}$.
- To construct RQ_l^*, $\mathcal{A}^*$ acts similarly to constructing Q_t^*. $\mathcal{A}^*$ also can determine which domain query tokens and user query tokens ask the same keyword with respect to $\Omega(q_1), \cdots, \Omega(q_t), \Omega(rq_1), \cdots, \Omega(rq_l)$. According to each $\Omega(rq_i) = \{did_i, did_{\mathcal{A}}, id(rs(rq_i))\}$ where i ranges from 1 to l, it computes rq_i^* as follows.

(a) $\mathcal{A}^*$ finds out $j \in [1, |DID|]$ such that $did'_j = did_i$, where $did'_j \in DID$. If no such j value exists, it computes $DID = DID \cup \{did_i\}$ and $j = |DID|$.

(b) If there exits $1 \leq k < i$ such that $id(rs(rq_k)) = id(rs(rq_i))$, which means rq_i and rq_k ask the same keyword, $\mathcal{A}^*$ sets $w_{i+t} = w_{k+t}$; otherwise, it proceeds to the next step.

(c) If there exits $1 \leq k \leq t$ such that $id(rs(q_k)) = id(rs(rq_i))$, which means rq_i and q_k ask the same keyword, $\mathcal{A}^*$ sets $w_{i+t} = w_k$; otherwise, it selects a random element $w_{i+t} \in \mathbb{G}_2^*$.

(d) $\mathcal{A}^*$ computes $x_j = select(j, X^*)$ and $s_j = select(j, S^*)$ as the querier domain's master secret key and query key, respectively. Then, it calculates $r_i^* = e\left(g_1, w_{i+t}\right)^{s_j^* x_j^*}, \sigma_i^* = \mathcal{H}(r_i^*)^{s_j^*}$ and sets $rq_i^* = (r_i^*, \sigma_i^*)$.

– To construct I^*, $\mathcal{A}^*$ initializes $I^* = \emptyset$. Then,

(a) For each $\Omega(q_i) = \{uid_i, did_i, id(rs(q_i))\}$, where $1 \leq i \leq t$, $\mathcal{A}^*$ knows the queried domain did_i and the underlying keyword w_i which is chosen for constructing q_i. If $did_i = did_{\mathcal{A}}$, it computes $I_i^* = e\left(g_1, w_i\right)^{x^*}$; otherwise, $\mathcal{A}^*$ computes $I_i^* = e\left(g_1, w_i\right)^{x_j^*}$, where j is the index of did_i in DID obtained via the operation in step (a) of constructing RQ_l^* with input did_i. If $I_i^* \notin I^*$, it computes $I^* = I^* \cup \{I_i^*\}$.

(b) For each $\Omega(rq_i)$, where $1 \leq i \leq l$, $\mathcal{A}^*$ computes $I_i^* = e\left(g_1, w_{i+t}\right)^{x^*}$, where w_{i+t} is the chosen keyword for constructing rq_i and x^* is regarded as $\mathcal{A}$ domain's master secret key. If $I_i^* \notin I^*$, it computes $I^* = I^* \cup \{I_i^*\}$.

(c) If $|I^*| < |I|$, $\mathcal{A}^*$ calculates the rest set as $I'^* = \{I_i^* : I_i^* \in_R \mathbb{G}_T^*\}_{i \in [|I^*|+1, |I|]}$, and concatenates it with I^* as $I^* = I^* \cup I'^*$.

– To construct RS_t^*, $\mathcal{A}^*$ sets up $RS_t^* = \emptyset$. Then, for each $\Omega(q_i)$ where $1 \leq i \leq t$, $\mathcal{A}^*$ sets $rs(q_i)^* = \emptyset$. Since $\mathcal{A}^*$ knows identities ids of ciphertext entries (e.t. $E(m)$) from $\Omega(q_i)$, it collects ciphertexts from ED^* whose identities are in ids and puts these ciphertexts into $rs(q_i)^*$. After that, $rs(q_i)^*$ is put into RS_t^*.

– Constructing RS'^*_l is the same as constructing RS^* excepting retrieving ids from $\Omega(rq_i)^*$, where i ranges from 1 to l. Each iteration produces a $rs(rq_i)^*$ with respective to i. Finally, $\mathcal{A}^*$ simulates $RS'^*_l = \{rs(rq_i)^*\}_{i\in[1,l]}$.

We show that V^* is computationally indistinguishable from V by comparing them component by component as follows.

– If $E(\cdot)$ is a pseudorandom permutation, ED^* and ED are computationally indistinguishable, since a real encrypted data $E(m)$ and a simulated $E(m)^* \in_R \{0,1\}^{|E(\cdot)|}$ are both random in the same domain $\{0,1\}^{|E(\cdot)|}$.

– For Q_t and Q_t^*, comparing an user query token generated by $\mathcal{A}$ as $q = \mathcal{H}(kw)^y$ and a simulated user query token generated by $\mathcal{A}^*$ as $q^* = w^{y^*}$, where $w \in_R \mathbb{G}_2^*$, q and q^* are computationally indistinguishable if $\mathcal{H}$ is a pseudorandom function, since both y_i and y_i^* are randomly chosen from $\mathbb{Z}_p^*$.

– For RQ_l and RQ_l^*, a domain query token generated by $\mathcal{A}$ as $(r = e(g_1, \mathcal{H}(kw))^{sx}, \sigma = \mathcal{H}(r)^s)$ is calculated from the random elements s, x in $\mathbb{Z}_p^*$. Similarly, a simulated domain query token generated by $\mathcal{A}^*$ as $(r^* = e(g_1, \mathcal{H}(w))^{s^*x^*}, \sigma = r^{*s^*})$ is also fielded from the random elements s^*, x^* in $\mathbb{Z}_p^*$. If $\mathcal{H}$ is a pseudorandom function, both r and r^* are random elements in the group $\mathbb{G}_T$, while both σ and σ^* are random elements in the group $\mathbb{G}_2^*$. Hence, a domain query token is computationally indistinguishable from a simulated domain query token.

– Comparing a real user search key $k = g_1^{\frac{x}{y}}$ and a simulated user search key $k^* = g_1^{\frac{x^*}{y^*}}$, they are computationally indistinguishable as y and y^* are random elements from $\mathbb{Z}_p^*$.

– For $PSKL$ and $PSKL^*$, an actual domain search key $(z = \frac{x}{x's'}, z' = g_1^{s'})$ is computationally indistinguishable to a simulated domain search key $(z^* = \frac{x^*}{x'^*s'^*}, z'^* = g_1^{s'^*})$, since all of x', s', x'^*, s'^* are random elements in $\mathbb{Z}_p^*$.

– For $PQKL$ and $PQKL^*$, a real domain query key s is computationally indistinguishable from a simulated domain query key s^*, since both s and s^* are randomly chosen from $\mathbb{Z}_p^*$.

– It is easy to see that I^* and I are computationally indistinguishable due to the elements of them being random in the group $\mathbb{G}_T$, which is exactly resulted from the pseudorandom function $\mathcal{H}$.

– Since the above components are computationally indistinguishable and the reply result sets are generated from those components, RS_t and RS_t^* are computationally indistinguishable. So do RS'^*_t and RS'_t.

Finally, based on the above indistinguishability results, the conclusion is yielded that A and A^* are computationally indistinguishable.

6.4.2 Query Unforgeability. Query unforgeability is defined for searchable encryptions in multi-user settings Bao et al. (2008). Particularly, M-EDESE is in a multi-domain setting, in which domains can be regarded as special users. Hence, we need to discuss the query unforgeability at both the user level and the domain level.

A user query token (or domain query token) is a user's legitimate query (or a domain's legitimate query) if it is indeed generated by the M-EDESE algorithm (search or upload process) with corresponding secret keys. Hence, in a nutshell, query unforgeability

is that for any user uid or domain did, no adversary is able to compute a user query token q and a domain query token rq without compromising the corresponding user query key QK_{uid} and the corresponding domain query key PQK_{did} respectively.

In the multi-user setting and the multi-domain setting of M-EDESE, we consider one type of adversaries and two types of victims respectively.

- **Adversary $\mathcal{A}$:** $\mathcal{A}$ is a collusion of malicious users and malicious domains (including their users, OCs, and KGCs).
- **Victim:** A victim is either $\mathcal{V}_{uid}$ or $\mathcal{V}_{did}$.
  - $\mathcal{V}_{uid}$ is an honest user whose identity is uid in a partial honest domain (KGC is fully trusted) whose identity is did.
  - $\mathcal{V}_{did}$ is an honest OC with fully trusted KGC in a domain did.

Adversary $\mathcal{A}$ can observe a set of information from malicious domains and their users. For a malicious domain did_i, $\mathcal{A}$ obtains $K_{did_i} = (did_i, UQKL', USKL, PSKL, PQKL, MK_{did_i})$, where $UPKL, PSKL, PQKL$ are the completed lists, and $UQKL'$ is a part of user query keys under domain did_i. If the domain did_i is partial honest, which means only its OC is malicious, we have $MK_{did_i} = \perp$. $\mathcal{A}$ also contains the collusion of malicious users under honest domains. $\mathcal{A}$ obtains QK_{uid_j} from such a malicious user whose identity is uid_j. In addition, $\mathcal{A}$ can observe all search and data upload interactions between all entities. Therefore, the acknowledge of $\mathcal{A}$ is defined as $\mathcal{K} = (\{K_{did_i}\}_{i\in[1,n]}, \{QK_{uid_j}\}_{j\in[1,m]})$ from n malicious domains and m malicious users.

Regarding two types of victims, two games are defined below. Both challengers of these two games simulate the execution of our M-EDESE and provide an oracle $\mathcal{O}$, which answers adversary $\mathcal{A}$'s queries on the executions of user enrollment, data uploading, collaboration establishment, and search with the acknowledge $\mathcal{K}$ of $\mathcal{A}$.

Game 1: $\mathcal{A}$ intends to forge a user query token q on behalf of $\mathcal{V}_{uid}$.

Game 2: $\mathcal{A}$ intends to forge a domain query token rq on behalf of $\mathcal{V}_{did}$.

We construct a BLS short signature from the above two games. The BLS setups with $(g_1, \mathcal{H}^*, e, x, g_1^x)$, where e is a bilinear map, g_1 is the generator of $\mathbb{G}_1$, $\mathcal{H}^*$ is a pseudorandom hash function $\mathcal{H}^* : \{0,1\}^* \rightarrow \mathbb{G}_2$, x is the secret signing key, and g_1^x is the corresponding public key. Lest $\mathcal{B}$ be a PPT adversary attempting to forge a short signature with respect to g_1^x. The proof requires demonstrating to construct the BLS short signature scheme, which provides the signing oracle $\mathcal{O}(x)$.

- **Game 1:** $\mathcal{A}$ observes a set of user query tokens $QT = \{\mathcal{H}(kw_1)^{y_{uid}}, \mathcal{H}(kw_2)^{y_{uid}}, \cdots\}$ from $\mathcal{V}_{uid}$ through data uploading and search oracle. It can also obtain $SK_{uid} = g_1^{\frac{x_{did}}{y_{uid}}}$. Since SK_{uid} has an unknown value x_{did} in the exponential of SK_{uid}, it is regarded as a random value. $\mathcal{B}$ regards QT as the BLS signatures under the signing key $x = y_{uid}$. Hence, $\mathcal{B}$ can simulate a BLS from the knowledge of $\mathcal{A}$ with $(g_1, \mathcal{H}^* = \mathcal{H}, e, x = y_{uid}, g_1^{y_{uid}})$.

- **Game 2:** $\mathcal{A}$ obtains a set of domain query tokens $RQ = \{(r_{kw_i} = e(g_1, \mathcal{H}(kw_i))^{s_{did}^{did_i} x_{did}}, \sigma_{kw_i} = \mathcal{H}(r_{kw_i})^{s_{did}^{did_i}})\}_{i \in [i,*]}$ from $\mathcal{V}_{did}$ through the search oracle. In addition, $\mathcal{A}$ obtains a set of $PSK_{did_i} = \{z_{did_i} = \frac{x_{did_i}}{x_{did} s_{did}^{did_i}}, z'_{did_i} = g_1^{s_{did}^{did_i}}\}$ from malicious domains. Hence, $\mathcal{A}$ can compute $r_{kw'} = (e(g_1, \mathcal{H}(kw'))^{x_{did_i}})^{\frac{1}{z_{did_i}}}$. However, $\mathcal{A}$ can not infer $\sigma_{kw'}$ since its exponential $s_{did}^{did_i}$ is unknown value. For σ_{kw_i}, $\mathcal{B}$ regards it as the BLS signature on the message $m = r_{kw_i}$ under the signing key $s_{did}^{did_i}$. Hence, $\mathcal{B}$ can simulate a BLS from the knowledge of $\mathcal{A}$ with $(g_1, \mathcal{H}^* = \mathcal{H}, e, x = s_{did}^{did_i}, g_1^{s_{did}^{did_i}}, m \in_R \mathbb{G}_T)$.

The simulation by $\mathcal{B}$ is perfect in Game 1 and Game 2. Hence, if $\mathcal{A}$ forges a legitimate user or domain query token based on a kw on behalf of $\mathcal{V}_{uid}$ or $\mathcal{V}_{did}$, she would also forge a valid BLS short signature on the kw.

6.4.3 Revocability. Revocability is a necessary requirement for any multi-user system. It is desirable to achieve user revocation and collaboration revocation in M-EDESE regarding its multi-user and multi-domain setting. Revocability enables an honest domain to revoke search abilities of its certain users and its certain PDs. Since the indistinguishability of indexes implies the fault of the searching process, it is effective to show revocability based on index indistinguishability. M-EDESE enables user revocability and collaboration revocability as defined below.

6.4.3.1 User Revocability. We construct the following game such that an adversary's advantage in attacking user revocability is winning the game. An adversary $\mathcal{A}$ performs in two phases. In phase 1, $\mathcal{A}$ acts as an authorized user and can access the search and data uploading oracle. At the end of phase 1, $\mathcal{A}$ chooses two never queried keywords kw'_1, kw'_2. The current knowledge of $\mathcal{A}$ contains $\{(kw_i, q_i, I_i)\}_{i\in[1,n]}$ and $\{kw'_1, kw'_2, QK\}$. After that, in phase 2, $\mathcal{A}$ is revoked and given I'_b based on one of kw'_1, kw'_2. $\mathcal{A}$ guess the value of b'. If $b = b'$, $\mathcal{A}$ wins the game.

Definition 14. *An M-EDESE system achieves user revocability if the below inequality holds for all PPT algorithms.*

$$
\begin{aligned}
|Pr[b' = b :& \\
& (GP, QKs, SKs, MKs) \leftarrow \textbf{Initial Setup\&Enrollment}; \\
& (\{(kw_i, q_i, I_i)\}_{i\in[1,n]}, kw'_1, kw'_2, QK) \leftarrow \textbf{Phase 1}_{\mathcal{O}}(\mathcal{A}); \\
& \textbf{Revoke}(\mathcal{A}); \\
& b \in_R \{1,0\}, I'_b \leftarrow \textbf{Search}(QK, kw'_b, SK); \\
& b' \leftarrow \mathcal{A}(\{(kw_i, q_i, I_i)\}_{i\in[1,n]}, kw'_1, kw'_2, QK) \\
] - \frac{1}{2}| &< \epsilon.
\end{aligned}
$$

6.4.3.2 Proof Sketch. To guess b' in the game, $\mathcal{A}$ needs to compute $I'_{kw'_1} = e(g_1, \mathcal{H}\ (kw'_1))^{x_{did'}}$. It is a hard discrete log problem as the exponential $x_{did'}$ is unknown for $\mathcal{A}$. As a consequence, from $\mathcal{A}$ perspective, $I'_{kw'_1}$ and $I'_{kw'_2}$ are independent of kw'_1 and kw'_2, respectively. Hence, the advantage of the adversary winning the game is negligible.

6.4.3.3 Collaboration Revocability. We construct the same game for collaboration revocability proof except that at the end of phase 1, the acknowledge of $\mathcal{A}$ is $\{(q_i, rq_i, I_i)\}_{i\in[1,n]}$ and $\{kw'_1, kw'_2, q'_1, q'_2, PQKL, USKL\}$.

Definition 15. *An M-EDESE system achieves collaboration revocability if the below inequality holds for all PPT algorithms.*

$$
\begin{aligned}
|\boldsymbol{Pr}[b' = b :\ & \\
& (GP, QKs, SKs, PQKs, SQKs, MKs) \leftarrow \textbf{Initial Setup\&Enrollment}; \\
& (\{(q_i, rq_i, I_i)\}_{i\in[1,n]}, q'_1, q'_2, kw'_1, kw'_2, PQKL, USKL) \leftarrow \textbf{Phase 1}_{\mathcal{O}}(\mathcal{A}); \\
& \textbf{Revoke}(\mathcal{A}); \\
& b \in_R \{1,0\}, I'_b \leftarrow \textbf{Search}(QK, kw'_b, SK, PQK_{did'}^{did''}, PSK_{did'}^{did''}); \\
& b' \leftarrow \mathcal{A}(\{(q_i, rq_i, I_i)\}_{i\in[1,n]}, q'_1, q'_2, kw'_1, kw'_2, PQKL, USKL) \\
] - \frac{1}{2}| < \epsilon.
\end{aligned}
$$

6.4.3.4 Proof Sketch. $\mathcal{A}$ can easily compute rq'_1 by calculating $r'_1 = e(SK, q'_1)^{s_{did'}^{did''}}$ and $\sigma'_1 = \mathcal{H}(r'_1)^{s_{did'}^{did''}}$. Recall that $I'_1 = {r'_1}^{z_{did'}^{did''}}$ and $z_{did'}^{did''} = \frac{x_{did''}}{x_{did'} s_{did'}^{did''}}$, it requires $z_{did'}^{did''}$ to compute I'_1 from rq'_1, which contains the unknown value $x_{did''}$. Similar to user revocability, from the $\mathcal{A}$'s perspective, I'_1 and I'_2 are independent of kw'_1 and kw'_2 respectively. Therefore, the advantage of the adversary winning the game is negligible.

6.5 Evaluations

In this section, we evaluate both the complexity and runtime performance of M-EDESE.

6.5.1 Complexities. We analyze the time complexities of data uploading processes on the user side and on the OC side. When a user uploads n keywords and an encrypted file, the time complexities of data uploading processes on both user and OC sides are $\mathcal{O}(n)$, which is independent of the number of users and domains. Specifically, the user side computes a hash function and an exponential operation in $\mathbb{G}_2$ for n times, while the OC side calculates a pairing operation for n times.

Then, we analyze the time complexities of search processes on the user side and on the OCs side. When a user queries a keyword, the time complexity of user query token generation on the user side is $\mathcal{O}(1)$ no matter how many domains are queried, while the time complexity of the search process on the OCs side is $\mathcal{O}(m)$, where m is the number of queried domains. OCs finally perform a pairing operation for $(2m + 1)$ times, an exponential operation for $(2m + 1)$ times, and a hash function for m times.

6.5.2 Experiments. We ran experiments on three algorithms, including user query generation, single domain index generation, and cross-domain index generation. We did not measure the time consumption on the CSP side since it is as fast as plaintext searching. In our experiments, we ran keyword search queries on a well-known opensource dataset, the Enron email corpus introduced by Enron Email Dataset (2019) and extracted a total of 5000 most frequent keywords from 30,109 emails.

We evaluated the performance of EDESE using different bilinear pairings, including SS512 (type A), MNT224 (type D), and BN256 (type F) pairings. All of them are equivalent to or exceed 80-bit security. We implemented M-EDESE based on Java programming language with version 1.8 and JPBC library implemented by De Caro and Iovino (2011) on a PC running 64-bit Windows 11 with 3.61 GHz Intel (R) Core (TM) i7-12700K and Memory 32GB RAM 3200MHz.

Table 15. Performance of query and index generations of M-EDESE in ms

Algorithm	SS512	MNT224	BN256
User query generation	4.72	14.25	4.96
Single domain index generation	5.69	14.29	20.43
Cross-domain index generation	4.96	70.7	93.24

From table 15, M-EDESE using SS512 type pairing runs faster than using the other two pairings. This is reasonable since SS512 is at the lowest security strength among these three types of pairings, while BN256 is at the highest security strength.

Considering that user query generation is performed by each individual user while both single domain index generation and cross-domain index generation are performed by OCs, the bottleneck of M-EDESE is the performance of OCs because each OC serves multiple users and performs index generations for all received user queries. In practice, such bottlenecks can be alleviated by leveraging powerful servers instead of PCs.

6.6 Conclusion

In this work, we proposed M-EDESE as a new secure keyword searching solution that is easily deployable and efficiently searchable in multi-domain settings. We presented a concrete construction of M-EDESE with formal security proofs. We also provided theoretical and experimental evaluations on M-EDESE showing that it is promising in practice.

With the design and evaluation of M-EDESE, this dissertation has addressed the spectrum of challenges in searchable encryption—from leakage resilience in symmetric settings, through decentralized trust in asymmetric constructions, to practical multi-domain deployment. The following chapter synthesizes these contributions, discusses their limitations, and outlines directions for future research.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

In this dissertation, we have addressed the critical challenge of building practical and secure searchable encryption systems for cloud-based data environments. As organizations increasingly outsource sensitive data to the cloud, the need to balance robust cryptographic protection with system deployability and decentralized trust becomes paramount. Our research has moved from establishing a theoretical foundation to designing leakage-resilient symmetric protocols, decentralizing trust in asymmetric frameworks, and finally realizing a multi-domain deployable system.

7.1 Summary of Research

The primary goal of this research was to bridge the gap between theoretical searchable encryption (SE) primitives and the practical requirements of modern cloud infrastructures. The contributions of this dissertation are summarized as follows:

1. **Systematic Taxonomy of Trust Models:** In Chapter II and our systematic study, we analyzed the paradigm shift from fully-trusted Private Key Generators (PKGs) to semi-trusted Key Curators (KCs). This work established the theoretical groundwork for Registration-Based Encryption (RBE) as a viable solution to the long-standing key escrow problem in searchable encryption.

2. **Mitigation of Volume Leakage:** In Chapter IV, we developed the **VLR-EDESE** scheme to fortify symmetric searchable encryption against Leakage-Abuse Attacks (LAAs). By introducing k-indistinguishability and obfuscation techniques, we demonstrated that query and document volume leakage can be significantly mitigated with substantially lower overhead than traditional ORAM-based solutions.

3. **Decentralized Expressive Search:** In Chapter V, we proposed **RBAE2KS**, which successfully integrated expressive query capabilities (monotone Boolean formulas) into a registration-based framework. This scheme effectively eliminates the reliance on a fully-trusted authority and provides robust defense against offline Keyword Guessing Attacks (KGA) while maintaining functional flexibility through Linear Secret Sharing Schemes (LSSS).

4. **Practical Multi-Domain Integration:** Finally, in Chapter VI, we realized **M-EDESE**, a system that extends SE to multi-domain environments. This system allows for cross-domain collaboration under inter-domain partnerships without requiring additional cryptographic support or cooperation from the cloud storage provider (CSP), achieving the goal of being "Easily Deployable."

7.2 Limitations

Despite the advancements presented in this work, certain limitations remain:

- **Communication Overhead:** While our schemes significantly improve performance over ORAM, the use of k-indistinguishability techniques inherently increases the storage footprint due to the introduction of dummy documents.

- **Dynamic Update Costs:** In the registration-based setting, maintaining and updating auxiliary information during frequent user joins or revocations remains a computationally intensive task for the Key Curator.

- **Expressiveness vs. Efficiency Trade-offs:** Although RBAE^{2}KS supports expressive queries, the underlying LSSS-based construction may not be optimal for all types of queries, and further optimizations are needed to balance expressiveness with efficiency.

- **Multi-Domain Scalability:** While M-EDESE enables multi-domain collaboration, the management of inter-domain partnerships and the potential for increased latency due to cross-domain queries require further investigation to ensure scalability in large deployments. Besides, inside of a domain, the system still requires a fully-trusted Key Generation Center.

7.3 Future Work

The findings of this dissertation open several promising avenues for future research:

- **Post-Quantum Security:** A critical next step is the transition of these protocols to post-quantum primitives. Migrating our pairing-based RBAE2KS construction to lattice-based cryptography will be essential for long-term security.
- **Hardware-Accelerated Search:** Investigating the integration of Trusted Execution Environments (TEEs), such as Intel SGX, to handle the Key Curator's registration and token transformation processes could further reduce trust assumptions and latency.
- **Fully Oblivious Systems:** Future research could explore lightweight oblivious techniques to hide not only volume but also the high-level multi-domain access patterns from the CSP.
- **Dynamic User Management:** Developing more efficient algorithms for handling dynamic user joins and revocations in the registration-based setting will be crucial for real-world deployments.
- **Decentralized Key Management:** Exploring decentralized approaches to key management in multi-domain settings that reduce reliance on a central trusted authority would enhance the robustness and scalability of the system.

7.4 Concluding Remarks

The journey toward "practical security" in cloud data systems is an iterative process of refining trust and optimizing performance. By addressing leakage, eliminating key escrow, and enabling multi-domain collaboration, this dissertation provides a scalable framework for the next generation of privacy-preserving search technologies.

REFERENCES CITED

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Computing Surveys (Csur)*, *51*(4), 1–35.

Agrawal, S., & Chase, M. (2017a). Fame: Fast attribute-based message encryption. In *Proceedings of the 2017 acm sigsac conference on computer and communications security* (p. 665–682). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/3133956.3134014` doi: 10.1145/3133956.3134014

Agrawal, S., & Chase, M. (2017b). Fame: Fast attribute-based message encryption. In *Proceedings of the 2017 acm sigsac conference on computer and communications security* (pp. 665–682).

Agrawal, S., & Chase, M. (2017c). Simplifying design and analysis of complex predicate encryption schemes. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 627–656).

Agrawal, S., Kumari, S., Yadav, A., & Yamada, S. (2023). Broadcast, trace and revoke with optimal parameters from polynomial hardness. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 605–636).

Akinyele, J. A., Garman, C., Miers, I., Pagano, M. W., Rushanan, M., Green, M., & Rubin, A. D. (2013). Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering*, *3*(2), 111–128. doi: 10.1007/s13389-013-0057-3

Akram, J., & Anaissi, A. (2024). Decentralized PKI framework for data integrity in spatial crowdsourcing drone services. In *2024 ieee international conference on web services (icws)* (pp. 643–653).

Al-Riyami, S. S., & Paterson, K. G. (2003). Certificateless public key cryptography. In *International conference on the theory and application of cryptology and information security* (pp. 452–473).

Amanatidis, G., Boldyreva, A., & O'Neill, A. (2007). Provably-secure schemes for basic query support in outsourced databases. In *Ifip annual conference on data and applications security and privacy* (pp. 14–30).

Angel, S., Chen, H., Laine, K., & Setty, S. (2018). PIR with compressed queries and amortized query processing. In *2018 ieee symposium on security and privacy (sp)* (pp. 962–979).

Arabnouri, A., & Shafieinejad, A. (2024). Bacase-sh: Blockchain-based authenticated certificate-less asymmetric searchable encryption for smart healthcare. *Peer-to-Peer Networking and Applications*, *17*(4), 2298–2314.

Attrapadung, N. (2014). Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 557–577).

Attrapadung, N., & Tomida, J. (2020). Unbounded dynamic predicate compositions in abe from standard assumptions. In *International conference on the theory and application of cryptology and information security* (pp. 405–436).

Attrapadung, N., & Tomida, J. (2024). A modular approach to registered abe for unbounded predicates. In *Annual international cryptology conference* (pp. 280–316).

Bao, F., Deng, R. H., Ding, X., & Yang, Y. (2008). Private query on encrypted data in multi-user settings. In *International conference on information security practice and experience (ispec)* (pp. 71–85).

Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., & Yang, K. (2001). On the (im) possibility of obfuscating programs. In *Annual international cryptology conference* (pp. 1–18).

Behrend, F. A. (1946). On sets of integers which contain no three terms in arithmetical progression. *Proceedings of the National Academy of Sciences*, *32*(12), 331–332.

Beimel, A. (1996a). Secure schemes for secret sharing and key distribution. *PhD thesis, Israel Institute of Technology, Technion*.

Beimel, A. (1996b). *Secure schemes for secret sharing and key distribution* (Unpublished doctoral dissertation). Technion – Israel Institute of Technology, Haifa, Israel. (Ph.D. Thesis)

Bellare, M., Boldyreva, A., Desai, A., & Pointcheval, D. (2001). Key-privacy in public-key encryption. In *International conference on the theory and application of cryptology and information security* (pp. 566–582).

Blackstone, L., Kamara, S., & Moataz, T. (2019). Revisiting leakage abuse attacks. *Cryptology ePrint Archive*.

Boneh, D., & Boyen, X. (2004). Efficient selective-id secure identity-based encryption without random oracles. In *International conference on the theory and applications of cryptographic techniques* (pp. 223–238).

Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004a). Public key encryption with keyword search. In *Advances in cryptology – eurocrypt 2004* (pp. 506–522). Springer.

Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004b). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506–522).

Boneh, D., & Franklin, M. (2001). Identity-based encryption from the weil pairing. In J. Kilian (Ed.), *Advances in cryptology — crypto 2001* (pp. 213–229). Berlin, Heidelberg: Springer Berlin Heidelberg.

Boneh, D., Lynn, B., & Shacham, H. (2001). Short signatures from the weil pairing. In *International conference on the theory and application of cryptology and information security (asiacrypt)* (pp. 514–532).

Boneh, D., Sahai, A., & Waters, B. (2011). Functional encryption: Definitions and challenges. In *Theory of cryptography: 8th theory of cryptography conference, tcc 2011, providence, ri, usa, march 28-30, 2011. proceedings 8* (pp. 253–273).

Boneh, D., Sahai, A., & Waters, B. (2012). Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, *55*(11), 56–64.

Boneh, D., & Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Theory of cryptography conference (tcc)* (pp. 535–554).

Bösch, C., Hartel, P., Jonker, W., & Peter, A. (2014). A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)*, *47*(2), 1–51.

Bost, R., & Fouque, P.-A. (2017). Thwarting leakage abuse attacks against searchable encryption–a formal approach and applications to database padding. *Cryptology ePrint Archive*.

Brakerski, Z., Lombardi, A., Segev, G., & Vaikuntanathan, V. (2018). Anonymous ibe, leakage resilience and circular security from new assumptions. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 535–564).

Branco, P., Lai, R. W. F., Maitra, M., Malavolta, G., Rahimi, A., & Woo, I. K. Y. (2025). Traitor tracing without trusted authority from registered functional encryption. In K.-M. Chung & Y. Sasaki (Eds.), *Advances in cryptology – asiacrypt 2024* (pp. 33–66). Singapore: Springer Nature Singapore.

Byun, J. W., Rhee, H. S., Park, H.-A., & Lee, D. H. (2006). Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Workshop on secure data management* (pp. 75–83). Springer.

Cai, J., Zhao, X., Li, D., Li, H., & Fan, K. (2025). Efficient and expressive public key authenticated encryption with keyword search in multi-user scenarios. *arXiv preprint arXiv:2503.16828*.

Cash, D., Grubbs, P., Perry, J., & Ristenpart, T. (2015). Leakage-abuse attacks against searchable encryption. In *Proceedings of the 22nd acm sigsac conference on computer and communications security* (pp. 668–679).

Cash, D., Jaeger, J., Jarecki, S., Jutla, C. S., Krawczyk, H., Rosu, M. C., & Steiner, M. (2014). Dynamic searchable encryption in very-large databases: data structures and implementation. In *Network and distributed system security symposium (ndss)* (Vol. 14, pp. 23–26).

Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Roşu, M.-C., & Steiner, M. (2013). Highly-scalable searchable symmetric encryption with support for boolean queries. In *Advances in cryptology–crypto 2013: 33rd annual cryptology conference. proceedings, part i* (pp. 353–373).

Chai, Q., & Gong, G. (2012). Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In *2012 ieee international conference on communications (icc)* (pp. 917–922).

Champion, J., Hsieh, Y.-C., & Wu, D. J. (2025). *Registered ABE and adaptively-secure broadcast encryption from succinct LWE.* Cryptology ePrint Archive, Paper 2025/044. Retrieved from `https://eprint.iacr.org/2025/044`

Chang, Y. C., & Mitzenmacher, M. (2005). Privacy preserving keyword searches on remote encrypted data. In *International conference on applied cryptography and network security (acns)* (pp. 442–455).

Chase, M., & Kamara, S. (2010). Structured encryption and controlled disclosure. In *Advances in cryptology-asiacrypt 2010: 16th international conference on the theory and application of cryptology and information security, singapore, december 5-9, 2010. proceedings 16* (pp. 577–594).

Chen, G., Lai, T.-H., Reiter, M. K., & Zhang, Y. (2018). Differentially private access patterns for searchable symmetric encryption. In *Ieee infocom 2018-ieee conference on computer communications* (pp. 810–818).

Chen, J., Gong, J., Kowalczyk, L., & Wee, H. (2018). Unbounded abe via bilinear entropy expansion, revisited. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 503–534).

Chen, J., Gong, J., & Wee, H. (2018). Improved inner-product encryption with adaptive security and full attribute-hiding. In *International conference on the theory and application of cryptology and information security* (pp. 673–702).

Cheng, L., & Meng, F. (2022). Public key authenticated encryption with keyword search from lwe. In *European symposium on research in computer security* (pp. 303–324).

Chiku, S., Hara, K., Hashimoto, K., Tomita, T., & Shikata, J. (2024). How to apply fujisaki-okamoto transformation to registration-based encryption. In *International conference on cryptology and network security* (pp. 145–165).

Cho, C., Döttling, N., Garg, S., Gupta, D., Miao, P., & Polychroniadou, A. (2017). Laconic oblivious transfer and its applications. In *Annual international cryptology conference* (pp. 33–65).

Chor, B., Fiat, A., & Naor, M. (1994). Tracing traitors. In *Annual international cryptology conference* (pp. 257–270).

Christopher, A., Dmitri, Z., Kiara, B., Kulpreet, S., Markus, S., & Will, A. (2017). *Decentralized public key infrastructure (dpki).* `https://github.com/WebOfTrustInfo`. (GitHub Repository; Accessed: June 6, 2025)

Chu, Q., Lin, L., Qian, C., & Chen, J. (2024). *Registered functional encryption for quadratic functions from MDDH.* Cryptology ePrint Archive, Paper 2024/177. Retrieved from `https://eprint.iacr.org/2024/177`

Cong, K., Eldefrawy, K., & Smart, N. P. (2021). Optimizing registration based encryption. In *Ima international conference on cryptography and coding* (pp. 129–157).

Cui, H., Wan, Z., Deng, R. H., Wang, G., & Li, Y. (2016). Efficient and expressive keyword search over encrypted data in cloud. *IEEE Transactions on Dependable and Secure Computing*, *15*(3), 409–422.

Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2006). Searchable symmetric encryption: improved definitions and efficient constructions. In *Proceedings of the 13th acm conference on computer and communications security* (pp. 79–88).

Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security*, *19*(5), 895–934.

Datta, P., & Pal, T. (2023). Registration-based functional encryption. *IACR Cryptol. ePrint Arch.*, *2023*, 457.

Datta, P., Pal, T., & Yamada, S. (2025). Registered FE beyond predicates: (attribute-based) linear functions and more. In K.-M. Chung & Y. Sasaki (Eds.), *Advances in cryptology – asiacrypt 2024* (pp. 65–104). Singapore: Springer Nature Singapore.

De Caro, A., & Iovino, V. (2011). jpbc: Java pairing based cryptography. In *2011 ieee symposium on computers and communications (iscc)* (pp. 850–855).

Döttling, N., & Garg, S. (2017a). Identity-based encryption from the diffie-hellman assumption. In J. Katz & H. Shacham (Eds.), *Advances in cryptology – crypto 2017* (pp. 537–569). Cham: Springer International Publishing.

Döttling, N., & Garg, S. (2017b). Identity-based encryption from the diffie-hellman assumption. In *Annual international cryptology conference* (pp. 537–569).

Döttling, N., Garg, S., Hajiabadi, M., & Masny, D. (2018). New constructions of identity-based and key-dependent message secure encryption schemes. In *Iacr international workshop on public key cryptography* (pp. 3–31).

Döttling, N., Kolonelos, D., Lai, R. W., Lin, C., Malavolta, G., & Rahimi, A. (2023). Efficient laconic cryptography from learning with errors. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 417–446).

Dua, A., Barpanda, S. S., Kumar, N., & Tanwar, S. (2020). Trustful: A decentralized public key infrastructure and identity management system. In *2020 ieee globecom workshops (gc wkshps* (pp. 1–6).

Dujmovic, J., Malavolta, G., & Qi, W. (2025). *Registration-based encryption in the plain model.* Cryptology ePrint Archive, Paper 2025/502. Retrieved from `https://eprint.iacr.org/2025/502`

El-Hajj, M., & Beune, P. (2024). Decentralized zone-based PKI: A lightweight security framework for iot ecosystems. *Information*, *15*(6), 304.

Elkin, M. (2010). An improved construction of progression-free sets. In *Proceedings of the twenty-first annual acm-siam symposium on discrete algorithms* (pp. 886–905).

Enron Email Dataset. (2019). *Enron email dataset.* `https://www.cs.cmu.edu/~./enron`. (Accessed: 2019-11-23)

Erdös, P., & Turán, P. (1936). On some sequences of integers. *Journal of the London Mathematical Society*, *1*(4), 261–264.

Escala, A., Herold, G., Kiltz, E., Ràfols, C., & Villar, J. (2017). An algebraic framework for diffie–hellman assumptions. *Journal of cryptology*, *30*(1), 242–288.

Faber, S., Jarecki, S., Krawczyk, H., Nguyen, Q., Rosu, M., & Steiner, M. (2015). Rich queries on encrypted data: Beyond exact matches. In *Computer security–esorics 2015: 20th european symposium on research in computer security, proceedings, part ii 20* (pp. 123–145).

Falzone, A. (2023, April). *Introducing dns-on-chain: A decentralized PKI solution using ethereum blockchain.* `https://github.com/AlexFalzone/DNSonChain`. (GitHub Repository; Accessed: June 6, 2025)

Feng, D., & Yang, K. (2022). Concretely efficient secure multi-party computation protocols: survey and more. *Security and Safety*, *1*, 2021001.

Fiat, A., & Naor, M. (1994). Broadcast encryption. In *Advances in cryptology—crypto'93: 13th annual international cryptology conference santa barbara, california, usa august 22–26, 1993 proceedings 13* (pp. 480–491).

Fiore, D., Kolonelos, D., & Perthuis, P. d. (2023). Cuckoo commitments: registration-based encryption and key-value map commitments for large spaces. In *International conference on the theory and application of cryptology and information security* (pp. 166–200).

Francati, D., Friolo, D., Maitra, M., Malavolta, G., Rahimi, A., & Venturi, D. (2023). Registered (inner-product) functional encryption. In *International conference on the theory and application of cryptology and information security* (pp. 98–133).

Freitag, C., Waters, B., & Wu, D. J. (2023). How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In *Annual international cryptology conference* (pp. 498–531).

Fromknecht, C., Velicanu, D., & Yakoubov, S. (2014). A decentralized public key infrastructure with identity retention. *Cryptology ePrint Archive*.

Garg, R., Lu, G., Waters, B., & Wu, D. J. (2024). Reducing the crs size in registered abe systems. In *Annual international cryptology conference* (pp. 143–177).

Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., & Waters, B. (2016). Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM Journal on Computing*, *45*(3), 882–929.

Garg, S., Gentry, C., Sahai, A., & Waters, B. (2013). Witness encryption and its applications. In *Proceedings of the forty-fifth annual acm symposium on theory of computing* (pp. 467–476).

Garg, S., Hajiabadi, M., Mahmoody, M., & Rahimi, A. (2018). Registration-based encryption: removing private-key generator from IBE. In *Theory of cryptography: 16th international conference, tcc 2018, panaji, india, november 11–14, 2018, proceedings, part i 16* (pp. 689–718).

Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A., & Sekar, S. (2019a). Registration-based encryption from standard assumptions. In *Iacr international workshop on public key cryptography* (pp. 63–93).

Garg, S., Hajiabadi, M., Mahmoody, M., Rahimi, A., & Sekar, S. (2019b). Registration-based encryption from standard assumptions. In *Iacr international workshop on public key cryptography* (pp. 63–93).

Garg, S., Mohassel, P., & Papamanthou, C. (2016). TWORAM: efficient oblivious RAM in two rounds with applications to searchable encryption. In *Annual international cryptology conference* (pp. 563–592).

Ge, C., Susilo, W., Baek, J., Liu, Z., Xia, J., & Fang, L. (2021). Revocable attribute-based encryption with data integrity in clouds. *IEEE Transactions on Dependable and Secure Computing*, *19*(5), 2864–2872.

George, M., Kamara, S., & Moataz, T. (2021). Structured encryption and dynamic leakage suppression. In A. Canteaut & F.-X. Standaert (Eds.), *Advances in cryptology – eurocrypt 2021* (pp. 370–396). Cham: Springer International Publishing.

Glaeser, N., Kolonelos, D., Malavolta, G., & Rahimi, A. (2023a). Efficient registration-based encryption. In *Proceedings of the 2023 acm sigsac conference on computer and communications security* (pp. 1065–1079).

Glaeser, N., Kolonelos, D., Malavolta, G., & Rahimi, A. (2023b). Efficient registration-based encryption. In *Proceedings of the 2023 acm sigsac conference on computer and communications security* (pp. 1065–1079).

Goyal, R., & Vusirikala, S. (2020). Verifiable registration-based encryption. In *Advances in cryptology–crypto 2020: 40th annual international cryptology conference, crypto 2020, santa barbara, ca, usa, august 17–21, 2020, proceedings, part i 40* (pp. 621–651).

Goyal, V., Lu, S., Sahai, A., & Waters, B. (2008). Black-box accountable authority identity-based encryption. In *Proceedings of the 15th acm conference on computer and communications security* (pp. 427–436).

Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006a). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th acm conference on computer and communications security* (p. 89–98). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1180405.1180418 doi: 10.1145/1180405.1180418

Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006b). Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th acm conference on computer and communications security* (pp. 89–98).

Gritti, C., Susilo, W., & Plantard, T. (2016). Certificate-based encryption with keyword search. *Journal of Internet Services and Information Security (JISIS)*, *6*(4), 1–34.

Gui, Z., Paterson, K. G., & Patranabis, S. (2023). Rethinking searchable symmetric encryption. In *2023 ieee symposium on security and privacy (sp)* (pp. 1401–1418).

Hajiabadi, M., Mahmoody, M., Qi, W., & Sarfaraz, S. (2023). Lower bounds on assumptions behind registration-based encryption. In *Theory of cryptography conference* (pp. 306–334).

Han, J., Yang, Y., Huang, X., Yuen, T. H., Li, J., & Cao, J. (2016). Accountable mobile e-commerce scheme via identity-based plaintext-checkable encryption. *Information Sciences*, *345*, 143–155.

He, D., Ma, M., Zeadally, S., Kumar, N., & Liang, K. (2017). Certificateless public key authenticated encryption with keyword search for industrial internet of things. *IEEE Transactions on Industrial Informatics*, *14*(8), 3618–3627.

He, K., Ma, S., & Wang, H. (2023). Multi-recipient public-key authenticated encryption with keyword search. In *International symposium on intelligence computation and applications* (pp. 287–296).

He, W., Akhawe, D., Jain, S., Shi, E., & Song, D. (2014). Shadowcrypt: Encrypted web applications for everyone. In *Proceedings of the 2014 acm sigsac conference on computer and communications security* (pp. 1028–1039).

Hoang, T., Yavuz, A. A., Durak, B. F., & Guajardo, J. (2017). Oblivious dynamic searchable encryption via distributed PIR and ORAM. *Cryptology ePrint Archive*.

Hobson, T., France, L., Greenbury, S., Hare, L., & Wochner, P. (2023). Trustchain–trustworthy decentralised public key infrastructure for digital credentials. In *Iet conference proceedings cp846* (Vol. 2023, pp. 31–40).

Hohenberger, S., Lu, G., Waters, B., & Wu, D. J. (2023). Registered attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 511–542).

Horwitz, J., & Lynn, B. (2002). Toward hierarchical identity-based encryption. In *International conference on the theory and applications of cryptographic techniques* (pp. 466–481).

Huang, Q., & Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, *403*, 1–14.

Hubacek, P., & Wichs, D. (2015). On the communication complexity of secure function evaluation with long output. In *Proceedings of the 2015 conference on innovations in theoretical computer science* (pp. 163–172).

Islam, M. S., Kuzu, M., & Kantarcioglu, M. (2012). Access pattern disclosure on searchable encryption: ramification, attack and mitigation. In *Ndss* (Vol. 20, p. 12).

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2016). Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kamara, S., & Moataz, T. (2019). Computationally volume-hiding structured encryption. In *Advances in cryptology–eurocrypt 2019: 38th annual international conference on the theory and applications of cryptographic techniques, darmstadt, germany, may 19–23, 2019, proceedings, part ii 38* (pp. 183–213).

Kamara, S., & Papamanthou, C. (2013). Parallel and dynamic searchable symmetric encryption. In *Financial cryptography and data security: 17th international conference, fc 2013, okinawa, japan, april 1-5, 2013, revised selected papers 17* (pp. 258–274).

Kamara, S., Papamanthou, C., & Roeder, T. (2012). Dynamic searchable symmetric encryption. In *Proceedings of the 2012 acm conference on computer and communications security* (pp. 965–976).

Kiayias, A., & Tang, Q. (2015). Making any identity-based encryption accountable, efficiently. In *Computer security–esorics 2015: 20th european symposium on research in computer security, vienna, austria, september 21-25, 2015, proceedings, part i 20* (pp. 326–346).

Kim, S., & Wu, D. J. (2020). Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In *International conference on the theory and application of cryptology and information security* (pp. 66–97).

Klimt, B., & Yang, Y. (2004). The Enron corpus: A new dataset for email classification research. In *Machine learning: Ecml 2004, 15th european conference on machine learning* (pp. 217–226). Springer. doi: 10.1007/978-3-540-30115-8_22

Koa, C.-G., Heng, S.-H., & Chin, J.-J. (2025). New ethereum-based distributed PKI with a reward-and-punishment mechanism. *Blockchain: Research and Applications*, *6*(1), 100239.

Lai, J., Deng, R. H., Guan, C., & Weng, J. (2013). Expressive CP-ABE with partially hidden access structures. In *Proceedings of the 8th ACM SIGSAC symposium on information, computer and communications security (asia ccs '13)* (pp. 19–30). ACM. doi: 10.1145/2484313.2484317

Lai, J., Deng, R. H., Zhao, Y., & Weng, J. (2013a). Accountable authority identity-based encryption with public traceability. In *Topics in cryptology–ct-rsa 2013: The cryptographers' track at the rsa conference 2013, san francisco, ca, usa, february 25-march 1, 2013. proceedings* (pp. 326–342).

Lai, J., Deng, R. H., Zhao, Y., & Weng, J. (2013b). Accountable authority identity-based encryption with public traceability. In *Cryptographers' track at the rsa conference* (pp. 326–342).

Lai, J., Zhou, X., Deng, R. H., Li, Y., & Chen, K. (2013). Expressive search on encrypted data. In *Proceedings of the 8th acm sigsac symposium on information, computer and communications security* (pp. 243–252).

Lambregts, S., Chen, H., Ning, J., & Liang, K. (2022). Val: Volume and access pattern leakage-abuse attack with leaked documents. In *European symposium on research in computer security* (pp. 653–676).

Langley, A. (2008). Crit-bit trees. *Retrieved January*, *18*, 2024.

Lau, B., Chung, S., Song, C., Jang, Y., Lee, W., & Boldyreva, A. (2014). Mimesis Aegis: A mimicry privacy shield-a system's approach to data privacy on public cloud. In *23rd usenix security symposium (usenix security 14)* (pp. 33–48).

Lee, H., Kim, K., & Lee, D. H. (2023). Accountable authority identity-based encryption with verifiable outsourced decryption. *Computers & Security*, *126*, 103073.

Lewko, A., Okamoto, T., Sahai, A., Takashima, K., & Waters, B. (2010a). Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 62–91).

Lewko, A., Okamoto, T., Sahai, A., Takashima, K., & Waters, B. (2010b). Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *Advances in cryptology — EUROCRYPT 2010* (Vol. 6110, pp. 62–91). Springer. doi: 10.1007/978-3-642-13190-5_4

Lewko, A., & Waters, B. (2011). Decentralizing attribute-based encryption. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 568–588).

Lewko, A. B., & Waters, B. (2011). Unbounded HIBE and attribute-based encryption. In *Advances in cryptology — EUROCRYPT 2011* (Vol. 6632, pp. 547–567). Springer. doi: 10.1007/978-3-642-20465-4_30

Li, H., Huang, Q., Shen, J., Yang, G., & Susilo, W. (2019). Designated-server identity-based authenticated encryption with keyword search for encrypted emails. *Information Sciences*, *481*, 330–343.

Libert, B., & Vergnaud, D. (2009). Towards black-box accountable authority ibe with short ciphertexts and private keys. In *Public key cryptography–pkc 2009: 12th international conference on practice and theory in public key cryptography, irvine, ca, usa, march 18-20, 2009. proceedings 12* (pp. 235–255).

Liu, C., Zhu, L., Wang, M., & Tan, Y.-a. (2014). Search pattern leakage in searchable encryption: Attacks and new construction. *Information Sciences*, *265*, 176–188.

Liu, H., Ming, Y., Wang, C., Zhao, Y., Zhang, S., & Lu, R. (2024). Blockchain-assisted verifiable certificate-based searchable encryption against untrusted cloud server for industrial internet of things. *Future Generation Computer Systems*, *153*, 97–112.

Liu, X., Li, H., Yang, G., Susilo, W., Tonien, J., & Huang, Q. (2019). Towards enhanced security for certificateless public-key authenticated encryption with keyword search. In *International conference on provable security* (pp. 113–129).

Liu, Z.-Y., Tseng, Y.-F., Tso, R., Chen, Y.-C., & Mambo, M. (2021). Identity-certifying authority-aided identity-based searchable encryption framework in cloud systems. *IEEE Systems Journal*, *16*(3), 4629–4640.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, *28*(2), 129–137.

Lu, Y., Li, J., & Wang, F. (2020). Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for iiots. *IEEE Transactions on Industrial Informatics*, *17*(4), 2696–2706.

Lu, Y., Li, J., & Zhang, Y. (2019). Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks. *IEEE Transactions on Services Computing*, *14*(6), 2041–2054.

Lv, Z., Hong, C., Zhang, M., & Feng, D. (2014). Expressive and secure searchable encryption in the public key setting (full version). *Cryptology ePrint Archive*.

Mahmoody, M., Qi, W., & Rahimi, A. (2022). Lower bounds for the number of decryption updates in registration-based encryption. In *Theory of cryptography conference* (pp. 559–587).

Meng, L., Chen, L., Tian, Y., Manulis, M., & Liu, S. (2024a). {FEASE}: Fast and expressive asymmetric searchable encryption. In *33rd usenix security symposium (usenix security 24)* (pp. 2545–2562).

Meng, L., Chen, L., Tian, Y., Manulis, M., & Liu, S. (2024b). {FEASE} : Fast and Expressive Asymmetric Searchable Encryption. In *Proceedings of the 33rd usenix security symposium (usenix security 24)* (pp. 2545–2562).

Meng, R., Zhou, Y., Ning, J., Liang, K., Han, J., & Susilo, W. (2017). An efficient key-policy attribute-based searchable encryption in prime-order groups. In *Provable security — 11th international conference, ProvSec 2017* (Vol. 10592, pp. 205–222). Springer. doi: 10.1007/978-3-319-68637-0_12

Mohassel, P. (2010). A closer look at anonymity and robustness in encryption schemes. In *International conference on the theory and application of cryptology and information security* (pp. 501–518).

Morrison, D. R. (1968). Patricia—practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM (JACM)*, *15*(4), 514–534.

Moser, L. (1950). *On sets of integers which contain no three in arithmetical progression and on sets of distances determined by finite point sets* (Unpublished doctoral dissertation). University of North Carolina at Chapel Hill.

Mu, Y., Varadharajan, V., & Quac Nguyen, K. (1999). Delegated decryption. In *Cryptography and coding: 7th ima international conference cirencester, uk, december 20–22, 1999 proceedings 7* (pp. 258–269).

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Naveed, M., Prabhakaran, M., & Gunter, C. A. (2014). Dynamic searchable encryption via blind storage. In *2014 ieee symposium on security and privacy* (pp. 639–654).

Ning, J., Huang, X., Poh, G. S., Yuan, J., Li, Y., Weng, J., & Deng, R. H. (2021). Leap: leakage-abuse attack on efficiently deployable, efficiently searchable encryption with partially known dataset. In *Proceedings of the 2021 acm sigsac conference on computer and communications security* (pp. 2307–2320).

Noroozi, M., & Eslami, Z. (2019). Public key authenticated encryption with keyword search: revisited. *IET Information Security*, *13*(4), 336–342.

Okamoto, T., & Takashima, K. (2016). Adaptively attribute-hiding (hierarchical) inner product encryption. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *99*(1), 92–117.

O'Neill, A. (2010). *Definitional issues in functional encryption.* Cryptology ePrint Archive, Paper 2010/556. Retrieved from `https://eprint.iacr.org/2010/556`

Oya, S., & Kerschbaum, F. (2022). IHOP: Improved statistical query recovery against searchable symmetric encryption through quadratic optimization. In *31st usenix security symposium (usenix security 22)* (pp. 2407–2424).

Pagh, R., & Rodler, F. F. (2004). Cuckoo hashing. *Journal of Algorithms*, *51*(2), 122–144.

Pakniat, N., Shiraly, D., & Eslami, Z. (2020). Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial iot. *Journal of Information Security and Applications*, *53*, 102525.

Patel, S., Persiano, G., Yeo, K., & Yung, M. (2019). Mitigating leakage in secure cloud-hosted data structures: Volume-hiding for multi-maps via hashing. In *Proceedings of the 2019 acm sigsac conference on computer and communications security* (pp. 79–93).

Peikert, C. (2009). Public-key cryptosystems from the worst-case shortest vector problem. In *Proceedings of the forty-first annual acm symposium on theory of computing* (pp. 333–342).

Poh, G. S., Chin, J. J., Yau, W. C., Choo, K. K. R., & Mohamad, M. S. (2017). Searchable symmetric encryption: designs and challenges. *ACM Computing Surveys (CSUR)*, *50*(3), 1–37.

Porwal, S., & Mittal, S. (2023). A review of key delegation schemes in ciphertext policy-attribute based encryption. In *2023 international conference on computational intelligence, communication technology and networking (cictn)* (pp. 309–314).

Pouliot, D., & Wright, C. V. (2016). The shadow nemesis: Inference attacks on efficiently deployable, efficiently searchable encryption. In *Proceedings of the 2016 acm sigsac conference on computer and communications security* (pp. 1341–1352).

Qin, B., Chen, Y., Huang, Q., Liu, X., & Zheng, D. (2020). Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Information Sciences*, *516*, 515–528.

Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, *56*(6), 1–40.

Riepel, D., & Wee, H. (2022a). Fabeo: Fast attribute-based encryption with optimal security. In *Proceedings of the 2022 acm sigsac conference on computer and communications security* (p. 2491–2504). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/3548606.3560699 doi: 10.1145/3548606.3560699

Riepel, D., & Wee, H. (2022b). Fabeo: Fast attribute-based encryption with optimal security. In *Proceedings of the 2022 acm sigsac conference on computer and communications security* (pp. 2491–2504).

Rivest, R. L., Adleman, L., Dertouzos, M. L., et al. (1978). On data banks and privacy homomorphisms. *Foundations of secure computation*, *4*(11), 169–180.

Sahai, A., & Seyalioglu, H. (2011). Fully secure accountable-authority identity-based encryption. In *International workshop on public key cryptography* (pp. 296–316).

Shamir, A. (1985). Identity-based cryptosystems and signature schemes. In *Advances in cryptology: Proceedings of crypto 84 4* (pp. 47–53).

Shen, C., Lu, Y., & Li, J. (2019). Expressive public-key encryption with keyword search: Generic construction from kp-abe and an efficient scheme over prime-order groups. *IEEE Access*, *8*, 93–103.

Shetty, J., & Adibi, J. (2004). The enron email dataset database schema and brief statistical report.. Retrieved from https://api.semanticscholar.org/CorpusID:59919272

Shiraly, D., Eslami, Z., & Pakniat, N. (2024). Hierarchical identity-based authenticated encryption with keyword search over encrypted cloud data. *Journal of Cloud Computing*, *13*(1), 112.

Shiraly, D., Pakniat, N., Noroozi, M., & Eslami, Z. (2022). Pairing-free certificateless authenticated encryption with keyword search. *Journal of Systems Architecture*, *124*, 102390.

Singh, A. K., Acharya, K., & Dutta, R. (2023). Cloud assisted semi-static secure accountable authority identity-based broadcast encryption featuring public traceability without random oracles. *Annals of Telecommunications*, *78*(1), 79–90.

Song, D. X., Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceeding 2000 ieee symposium on security and privacy. s&p 2000* (pp. 44–55).

Springer, J., & Haindl, P. (2024). Blockchain-based PKI within a corporate organization: advantages and challenges. *arXiv preprint arXiv:2407.04536*.

Stefanov, E., Papamanthou, C., & Shi, E. (2013). Practical dynamic searchable encryption with small leakage. *Cryptology ePrint Archive*.

Taha, M., Zhong, T., Elhabob, R., Xiong, H., Kumari, S., Chen, C.-M., & Alenazi, M. J. (2025). Identity-based searchable encryption with cryptographic reverse firewalls for iot-based healthcare systems. *Computers and Electrical Engineering*, *124*, 110404.

Tallón-Ballesteros, A. (2022). A new identity-based encryption scheme with accountable authority based on sm9. In *Proceedings of cecnet 2021: The 11th international conference on electronics, communications and networks (cecnet), november 18-21, 2021* (Vol. 345, p. 440).

Tseng, Y.-F., Fan, C.-I., & Liu, Z.-C. (2022). Fast keyword search over encrypted data with short ciphertext in clouds. *Journal of Information Security and Applications*, *70*, 103320.

Wang, G., Liu, Q., & Wu, J. (2010). Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *Proceedings of the 17th acm conference on computer and communications security* (pp. 735–737).

Wang, Q., Li, R., Wang, Q., Galindo, D., Chen, S., & Xiang, Y. (2023). Transparent registration-based encryption through blockchain. *Distributed Ledger Technologies: Research and Practice*, *2*(1), 1–14.

Waters, B. (2011). Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *International workshop on public key cryptography* (pp. 53–70).

Wee, H. (2022). Optimal broadcast encryption and cp-abe from evasive lattice assumptions. In O. Dunkelman & S. Dziembowski (Eds.), *Advances in cryptology – eurocrypt 2022* (pp. 217–241). Cham: Springer International Publishing.

Wood, G., et al. (2014). Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, *151*(2014), 1–32.

Wu, Z., & Li, R. (2023). OBI: a multi-path oblivious RAM for forward-and-backward-secure searchable encryption. In *Ndss.*

Xu, P., Jin, H., Wu, Q., & Wang, W. (2012). Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Transactions on computers*, *62*(11), 2266–2277.

Xu, Q., Zhang, Q., Yu, B., Shi, N., Wang, C., & He, W. (2021). Decentralized and expressive data publish-subscribe scheme in cloud based on attribute-based keyword search. *Journal of Systems Architecture*, *119*, 102274.

Yang, L., Liu, J., & Li, G. (2022). Black-box accountable authority CP-ABE scheme for cloud-assisted e-health system. *Wireless Communications and Mobile Computing*, *2022*, 1-12.

Yang, X., Chen, G., Wang, M., Li, T., & Wang, C. (2020). Multi-keyword certificateless searchable public key authenticated encryption scheme based on blockchain. *IEEE Access*, *8*, 158765–158777.

Yau, W.-C., Heng, S.-H., & Goi, B.-M. (2008). Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In *International conference on autonomic and trusted computing* (pp. 100–105). Springer.

Yin, H., Zhang, J., Xiong, Y., Ou, L., Li, F., Liao, S., & Li, K. (2019). Cp-abse: A ciphertext-policy attribute-based searchable encryption scheme. *IEEE Access*, *7*, 5682–5694.

Yuan, J., Li, Y., Ning, J., & Deng, R. H. (2022). M-edese: Multi-domain, easily deployable, and efficiently searchable encryption. In *International conference on information security practice and experience* (pp. 606–623).

Zhang, L., Yang, G., Song, C., & Wu, Q. (2023). Accountable multi-authority attribute-based data access control in smart grids. *Journal of King Saud University-Computer and Information Sciences*, *35*(7), 101597.

Zhang, Y., Chen, J., He, D., & Zhang, Y. (2025). Bounded collusion-resistant registered functional encryption for circuits. In K.-M. Chung & Y. Sasaki (Eds.), *Advances in cryptology – asiacrypt 2024* (pp. 32–64). Singapore: Springer Nature Singapore.

Zhang, Y., Katz, J., & Papamanthou, C. (2016). All your queries are belong to us: the power of file-injection attacks on searchable encryption. In *25th usenix security symposium (usenix security 16)* (pp. 707–720).

Zhang, Y., Wen, L., Zhang, Y., & Wang, C. (2019). Designated server certificateless deniably authenticated encryption with keyword search. *IEEE Access*, *7*, 146542–146551.

Zhang, Y., Zhao, J., Zhu, Z., Gong, J., & Chen, J. (2024). Registered attribute-based signature. In *Iacr international conference on public-key cryptography* (pp. 133–162).

Zhao, F., Weng, J., Xie, W., Hou, L., & Li, M. (2024). Time-based attribute-based proxy re-encryption with decryption key update. *Designs, Codes and Cryptography*, *92*(12), 4099–4129.

Zhao, Z., Guo, F., Lai, J., Susilo, W., Wang, B., & Hu, Y. (2020). Accountable authority identity-based broadcast encryption with constant-size private keys and ciphertexts. *Theoretical Computer Science*, *809*, 73–87.

Zhao, Z., Wu, G., Guo, F., Susilo, W., Mu, Y., Wang, B., & Hu, Y. (2020). Black-box accountable authority identity-based revocation system. *The Computer Journal*, *63*(4), 525–535.

Zhu, Z., Li, J., Zhang, K., Gong, J., & Qian, H. (2024). Registered functional encryptions from pairings. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 373–402).

Zhu, Z., Zhang, K., Chen, Z., Gong, J., & Qian, H. (2025). *Black-box registered ABE from lattices.* Cryptology ePrint Archive, Paper 2025/051. Retrieved from `https://eprint.iacr.org/2025/051`

Zhu, Z., Zhang, K., Gong, J., & Qian, H. (2023). Registered abe via predicate encodings. In *International conference on the theory and application of cryptology and information security* (pp. 66–97).