A Survey on Secure Data Chunking and Encrypted Metadata Indexing in Cloud-Based system

Mohanaprakash T A [1] , Upendra Kumar [2]

[1]Research Scholar , Lincoln Global Postdoctoral Research (LGPR) , Lincoln University College, Malaysia
pdf.mohanaprakash@lincoln.edu.my
[2]Institute of Engineering and Technology , Lucknow , India
Adjunct Research faculty , Lincoln Global Postdoctoral Research (LGPR), Lincoln University College, Malaysia

upendra.ietlko@gmail.com

Abstract: The adoption of cloud platforms for bibliographic control systems introduces significant data privacy challenges, as sensitive metadata and content must be secured from untrusted cloud providers. While traditional encryption protects data at rest, it inhibits essential processing functions like search and data management. This survey comprehensively examines the potential of Homomorphic Encryption (HE) to enable secure computations on encrypted data, specifically focusing on its application for privacy-preserving data chunking and encrypted metadata indexing. This paper analyze the taxonomy of HE schemes (FHE, SHE, PHE), their performance trade-offs, and relevant algorithmic adaptations for bibliographic operations. Furthermore, the paper identifies persistent challenges—including computational overhead, efficient secure chunking, and integration with standards like MARC—and outlines a roadmap for future research to realize practical, secure cloud-based bibliographic systems.

Keywords: Homomorphic Encryption , Data Chunking , Cloud Computing , Metadata , Data Security

1. Introduction

Bibliographic control systems form the backbone of global knowledge organization, encompassing libraries, digital archives, and citation databases.[17] These systems rely on rich, structured metadata (e.g., MARC records) to describe, discover, and manage intellectual assets. The cloud presents a compelling platform for these systems, offering elastic storage and computational resources. However, entrusting sensitive or proprietary bibliographic data to a cloud service provider (CSP) creates a significant privacy risk. While encryption is a standard mitigation[1], it traditionally creates a trade-off between security and functionality: data must be decrypted on the server to be processed, exposing it to the CSP. Homomorphic Encryption (HE)[6] offers a potential solution to this paradox. HE allows for specific algebraic operations to be performed directly on ciphertexts, generating an encrypted result that, when

decrypted, matches the result of operations performed on the plaintext. For cloud-based bibliographic systems, this enables two transformative capabilities:

Secure Data Chunking: Large text-based resources (e.g., entire e-books, journal issues) can be encrypted and then logically segmented into smaller, encrypted chunks (e.g., by chapter or page) on the cloud server for efficient storage and retrieval, all without the CSP ever accessing the cleartext[2].

Encrypted Metadata Indexing: An encrypted index can be built from encrypted metadata records. Users can then submit encrypted search queries, which the CSP can execute over the encrypted index, returning relevant encrypted records without learning the query or the contents of the index[3].

This paper surveys the intersection of these advanced cryptographic techniques with the practical demands of modern bibliographic systems. We explore the current state of the art, synthesize key approaches, and highlight the pressing research challenges that must be overcome to realize truly secure and functional cloud-based bibliographic control.[12]

2: Survey of Homomorphic Encryption for Bibliographic Systems

2.1 Introduction to the Survey

The primary goal of this chapter is to survey the current landscape of Homomorphic Encryption (HE) technologies, with a dedicated focus on their application to two critical tasks in cloud-based bibliographic systems: secure data chunking and encrypted metadata indexing. It aims to provide a comprehensive overview of the available cryptographic tools, their capabilities, and their practical suitability for enabling these privacy-preserving operations.

The chapter is structured as follows: It begins by establishing a foundational understanding of the spectrum of HE schemes, from Partial (PHE) to Somewhat (SHE) and Fully Homomorphic Encryption (FHE). It then delves into a detailed analysis of HE's role in secure data chunking, examining both size-based and the challenging content-aware methods. Following this, the chapter explores the construction and querying of encrypted metadata indexes for bibliographic search. Finally, it concludes with a review of practical performance considerations and the existing software ecosystem that supports HE development.

2.2 Foundational Homomorphic Encryption Schemes

2.2.1 Partially Homomorphic Encryption (PHE):

Partially Homomorphic Encryption (PHE) is defined by its capability to perform an unlimited number of operations, but of only a single type—either addition or multiplication—on ciphertexts. Its principle advantage lies in this singular focus, which allows for designs that are vastly more computationally efficient than their full homomorphic counterparts. The seminal Paillier cryptosystem is the foremost example of an additively homomorphic scheme, where the product of two ciphertexts decrypts to the sum of their corresponding plaintexts. In contrast, the widely known RSA algorithm is multiplicatively

homomorphic. Within bibliographic systems, this targeted functionality makes PHE highly practical for specific, well-defined tasks. It is perfectly suited for the secure tallying and aggregation of data, such as calculating the cumulative size of encrypted data chunks for storage management or performing simple keyword frequency counts across a collection without decrypting any individual record. Furthermore, its efficiency enables privacy-preserving access logging, where user interactions with sensitive materials can be recorded and audited in an encrypted form, ensuring both accountability and confidentiality.[6]

2.2.2 Somewhat Homomorphic Encryption (SHE):

Somewhat Homomorphic Encryption (SHE) represents a critical middle ground in the cryptographic landscape. Its core principle is the support for a limited number of both addition and multiplication operations, often described as evaluating low-depth arithmetic circuits. This balanced capability is achieved by managing the inherent "noise" growth within the ciphertext; once a certain threshold of operations is passed, the noise overwhelms the data, making decryption impossible. Prominent schemes like BGV and BFV are architected around this principle, providing a practical toolkit for computations that are more complex than simple arithmetic but do not require the unbounded flexibility of FHE.[6]

For bibliographic systems, this makes SHE uniquely relevant. It strikes a vital balance between functionality and performance, being suitable for constructing encrypted searchable indexes from metadata. A library could homomorphically build an inverted index by processing encrypted keywords and document identifiers. Furthermore, SHE is capable of executing constrained search queries directly on these encrypted indexes. A user could, for instance, submit an encrypted conjunctive query (e.g., "author X AND keyword Y"), and the cloud server could process it without decryption. This enables meaningful, private search functionality while avoiding the prohibitive computational overhead associated with Fully Homomorphic Encryption, making SHE a pragmatic candidate for real-world, privacy-preserving bibliographic search and discovery.[7]

2.2.3 Fully Homomorphic Encryption (FHE):

Fully Homomorphic Encryption (FHE) stands as the pinnacle of secure computation, governed by the principle of supporting arbitrary computations—comprising unlimited additions and multiplications—directly on ciphertexts. This is achieved through sophisticated "bootstrapping" techniques that periodically reduce computational noise, enabling theoretically limitless evaluation of any function. A diverse ecosystem of key schemes exists to cater to different data types and performance needs. BGV and BFV excel with exact integer arithmetic, while CKKS is revolutionary for its efficient handling of approximate floating-point numbers, crucial for real-world data analysis. Conversely, FHEW and TFHE optimize for gate-based bootstrapping, offering faster performance for binary circuit evaluation.[4]

For bibliographic systems, FHE represents the ultimate "end goal," unlocking the most advanced functionalities on encrypted data. It is the only technology that can enable complex, privacy-preserving operations such as ranked keyword search—where results are returned by relevance using algorithms like TF-IDF—direct semantic analysis of text, or intricate management of encrypted data chunks. The CKKS scheme, in particular, is a transformative tool, as it allows for machine learning and advanced analytics to be performed directly on encrypted metadata, facilitating tasks like trend analysis, personalized recommendation, and automated classification without ever compromising user confidentiality.[5]

2.3 Homomorphic Encryption for Secure Data Chunking

2.3.1 The Chunking Problem in Bibliographic Context

In cloud-based bibliographic systems, efficiently managing large text objects—such as digitized books, extensive journal articles, or research datasets—is a fundamental challenge. These items are too sizable to store, transfer, or process as single monolithic files. The standard solution is data chunking: splitting them into smaller, logically segmented, and manageable pieces (e.g., by chapter, section, or fixed size). This approach optimizes storage utilization, enables parallel processing, and facilitates efficient caching and retrieval. However, in a privacy-centric model where the cloud provider is untrusted, this chunking must occur *after* the data is encrypted to prevent exposure of sensitive content. This requirement transforms a straightforward storage problem into a complex cryptographic one, necessitating methods to logically partition data while it remains in an encrypted state, entirely opaque to the server performing the operation.

2.3.2 Size-Based Chunking with PHE

Partially Homomorphic Encryption (PHE), specifically additively homomorphic schemes like Paillier, offers a practical and highly efficient solution for one type of chunking: size-based segmentation. The process operates on encrypted data without decryption. First, the client encrypts each fixed-size block of the plaintext and uploads the sequence of ciphertexts. To create chunks of a predetermined target size (e.g., 10MB), the server can homomorphically add the encrypted sizes of consecutive blocks. Since the Paillier scheme possesses the additive property, the product of these individual size ciphertexts decrypts to the sum of their plaintext sizes. The cloud server can therefore calculate the cumulative size of blocks and logically group them into larger chunks once the homomorphically computed sum nears the target threshold, all without knowing the actual sizes or content. This method provides a balance between practicality and privacy for basic storage optimization.[2]

2.3.3 Content-Aware Chunking (A Major Challenge)

While size-based chunking is efficient, content-aware chunking—segmenting data at logical boundaries like chapter headings, section breaks, or semantic shifts—remains a formidable challenge under encryption. This process inherently requires pattern matching, a operation notoriously expensive under FHE. Identifying a string pattern like "CHAPTER X" within an encrypted text involves performing a complex sequence of homomorphic comparisons across the entire dataset for every potential match, a process with multiplicative depth that quickly becomes computationally intractable for large texts. Early research into this problem explores optimized string matching algorithms, such as the Single Pattern Hamming Distance (SPHD) algorithm, adapted for the encrypted domain. These methods represent significant theoretical advances but are currently hampered by prohibitive computational overhead, rendering them impractical for real-world, large-scale bibliographic applications and marking this area as a key frontier for future research.

2.3.4 Secure Deduplication

Secure deduplication[10] is a critical storage optimization technique for identifying and eliminating redundant data chunks across a vast encrypted bibliographic[7] repository. Homomorphic Encryption provides a mechanism to achieve this without compromising privacy. The core idea is to compute a deterministic, homomorphic fingerprint or hash of an encrypted data chunk. Using an additively homomorphic scheme, the server can calculate the Hamming distance or another similarity metric

between the encrypted fingerprints of different chunks[15]. If the homomorphic operation on two ciphertexts yields an encrypted result that decrypts to zero (or a value below a threshold), it indicates the original plaintext chunks are identical or very similar, enabling the system to store only one copy and create pointers to it. This process allows the cloud provider to significantly reduce storage costs and improve efficiency while remaining completely oblivious to the actual content of the duplicated chunks.[4,16]

2.4 Homomorphic Encryption for Encrypted Metadata Indexing

2.4.1 Index Structures

Traditional information retrieval relies on efficient index structures like the inverted index, which maps keywords to the list of documents containing them. Adapting these for the encrypted domain is a complex but vital task. An encrypted inverted index cannot simply store plaintext lists; instead, each posting list—the set of document identifiers for a keyword—must itself be encrypted, often using homomorphic encryption to allow for future computations. Beyond inverted indexes, probabilistic data structures like Bloom filters are also adapted. An encrypted Bloom filter can homomorphically test for set membership, allowing a server to check if a specific encrypted keyword is *likely* in a document without knowing the keyword itself. These encrypted data structures form the foundational layer upon which private search is built, but they introduce significant overhead in terms of storage size and computational complexity compared to their plaintext counterparts.

2.4.2 Index Construction under HE

Building an encrypted index from encrypted metadata is a non-trivial process that occurs on the client-side or requires client-aided computation. The core challenge involves processing a collection of encrypted document records—each containing encrypted fields for title, author, and keywords—and homomorphically generating the encrypted index entries. This process involves performing homomorphic comparisons to categorize and sort encrypted terms. For instance, to build an inverted index, the client must algorithmically determine that multiple documents contain the same encrypted keyword and then homomorphically construct a single encrypted posting list that aggregates those document IDs. This requires a series of equality checks and data structuring operations (like appending to a list) to be performed entirely on ciphertexts, making index construction a one-time but computationally intensive prerequisite for enabling efficient future queries.

2.4.3 Query Processing

Once an encrypted index is built, processing queries over it involves several methods. For **exact keyword search**, the client encrypts their search term and sends it to the server, which performs a homomorphic comparison against encrypted index terms, returning the matching encrypted posting list for Boolean retrieval. **Ranked search** is more advanced, leveraging schemes like CKKS that support approximate arithmetic on encrypted vectors. Here, the index stores encrypted term frequency-inverse document frequency (TF-IDF) vectors for documents. The server can homomorphically compute the cosine similarity between an encrypted query vector and these document vectors, resulting in an encrypted relevance score. The server can then return documents ordered by these scores without decrypting them. For **range queries** on numerical fields like publication year, the server can homomorphically evaluate whether an encrypted value falls within a given encrypted range, enabling private filtering.

2.4.5 Hybrid Approaches

Recognizing the performance limitations of pure HE solutions, hybrid cryptographic approaches have emerged as the most pragmatic path forward. These systems combine the extreme efficiency of Searchable Symmetric Encryption (SSE) for fast, filter-based retrieval with the powerful computational capabilities of HE. In a typical hybrid model, SSE is used as a first step to quickly and efficiently retrieve a broad superset of encrypted documents that match a query based on keywords. Then, a more computationally expensive homomorphic operation is applied only to this much smaller subset of results. For example, HE can be used to perform private ranking on the shortlisted results from SSE, calculating precise relevance scores (TF-IDF) or performing complex comparisons.[22] This strategy effectively mitigates performance issues by limiting the use of costly homomorphic operations to a refined dataset, making complex private search feasible for large-scale bibliographic databases.

2.5 Performance Considerations and Practical Tools

2.5.1 Computational Overhead

The primary impediment to the widespread adoption of Homomorphic Encryption is its immense computational overhead. Published benchmarks reveal that homomorphic operations are orders of magnitude slower than their plaintext equivalents. A single multiplication of ciphertexts can be thousands of times slower, and the bootstrapping operation required for unlimited computation (FHE) can take seconds or even minutes per ciphertext.[21] For bibliographic operations, this translates to query latencies that are currently impractical for interactive systems. Building an encrypted index on a large dataset could take days, and executing a complex ranked search query over a large collection could require minutes of server-time. This overhead is the central trade-off, forcing a careful analysis of which operations are essential and must be performed homomorphically versus those that can be handled through more efficient means.

2.5.2 Ciphertext Expansion

A second critical overhead is ciphertext expansion, where encrypted data is significantly larger than the original plaintext. Depending on the chosen security parameters and scheme, a single bit or value of plaintext can balloon into a ciphertext that is kilobytes in size, leading to expansion factors of 1000x or more. This has a direct impact on both storage and bandwidth requirements. Storing an encrypted bibliographic index or a collection of encrypted chunks can require terabytes of space where gigabytes once sufficed. Furthermore, transmitting these large ciphertexts between the client and the cloud server consumes substantial bandwidth, adding latency to every interaction and increasing operational costs. This expansion is a fundamental property of current lattice-based cryptosystems and must be factored into any system design.[19]

2.5.3 Optimization Techniques

To mitigate these overheads, several key optimization techniques are employed. **Batching** is arguably the most important. It allows a single ciphertext to encode a vector of hundreds or thousands of plaintext values using polynomial Chinese Remainder Theorem (CRT) packing. This enables Single Instruction, Multiple Data (SIMD) operations, where one homomorphic operation (e.g., addition) is simultaneously performed on all values in the vector, dramatically improving throughput and amortizing costs. **Parameter selection** is another crucial lever. Cryptographic parameters (e.g., polynomial ring dimension, ciphertext

modulus) must be carefully chosen to provide sufficient security while minimizing performance penalties. There is a constant triage between security level, computational depth (number of operations supported), and performance, requiring tailored choices for each specific bibliographic task.[11]

2.5.4 Available Libraries

The practical implementation of HE is facilitated by several open-source libraries. **Microsoft SEAL** is one of the most popular and well-documented, offering implementations of the BFV (for integers) and CKKS (for approximate numbers) schemes, making it highly suitable for ranked search and analytics on numerical metadata. **PALISADE** is another versatile library that supports multiple schemes (BGV, BFV, CKKS, FHEW, TFHE), providing developers with flexibility for different use cases. **HElib** is an older, highly optimized library primarily for the BGV scheme. **TFHE** is specialized for gate-based bootstrapping and offers very fast evaluation of binary circuits. For bibliographic applications requiring arithmetic on integers or real numbers (e.g., TF-IDF), SEAL and PALISADE's CKKS implementations are often the most relevant starting points.

3: Key Challenges and Limitations

3.1 Introduction to Challenges

While the theoretical promise of Homomorphic Encryption is transformative for cloud-based bibliographic control, its path to widespread adoption is obstructed by a series of profound practical hurdles. This chapter moves beyond the potential of HE to ground the discussion in the stark realities that currently impede its deployment. It provides a critical examination of the performance bottlenecks, algorithmic limitations, system integration barriers, and lingering security concerns that collectively define the significant gap between cryptographic theory and operational practice.[16]

3.2 Performance and Scalability Bottlenecks

The most immediate and daunting challenges are performance-related.[17] **Computational Latency** remains the primary bottleneck; homomorphic operations, particularly multiplications and the bootstrapping necessary for unlimited computation, are orders of magnitude slower than plaintext processing. This makes real-time querying of large-scale bibliographic databases utterly infeasible, transforming searches that would take milliseconds into processes requiring minutes or hours. **Storage Overhead** presents a parallel economic challenge; the massive ciphertext expansion inherent to lattice-based cryptography can inflate storage requirements by factors of 1,000x or more. Storing terabytes of encrypted data where gigabytes once sufficed creates a prohibitive cost model for cloud storage. Furthermore, this expansion drives a significant **Communication Overhead**, as moving these large ciphertexts between client and server consumes immense bandwidth, increasing latency and operational expenses for every interaction.[5]

3.3 Algorithmic and Functional Challenges

Beyond raw performance, fundamental algorithmic gaps exist. [18]**Secure and Efficient Data**

Chunking highlights a key functional trade-off: while size-based chunking with PHE is efficient, content-aware chunking (e.g., by chapter) requires complex pattern matching under FHE that is currently computationally intractable for large texts. **Complex Query Semantics** represent another major frontier; supporting the advanced search features expected by modern users—such as proximity search, phrase search, or faceted navigation—involves computational circuits of such depth and complexity that they are effectively impossible with current technology. **Dynamic Data Management** is also a critical unsolved problem; efficiently updating an encrypted index to insert a new record or delete an existing one without performing a full, costly re-encryption and re-indexing of the entire database remains a significant challenge.[2]

3.4 System Integration and Standardization Hurdles

Integrating HE into existing bibliographic ecosystems presents its own set of obstacles. **Integration with Bibliographic Standards** involves the non-trivial task of mapping HE's mathematical operations onto the rich, hierarchical, and often complex structures of standards like MARC21 and BIBFRAME. The **Lack of Standardized APIs** forces application developers to possess deep cryptographic expertise, as they must work directly with low-level library functions instead of high-level, domain-specific instructions for search or indexing. This drastically slows development and adoption. Finally, **Key Management** introduces a critical practical problem: securely generating, storing, rotating, and distributing the sensitive secret keys in a multi-user environment like a library consortium, without creating a single point of failure, is a complex infrastructural challenge distinct from the cryptography itself.[19]

3.5 Security and Privacy Considerations

Even if performance and integration hurdles are overcome, subtle security risks persist. **Side-Channel Attacks** pose a persistent threat; while the data itself is encrypted, the *access patterns*—such as which specific encrypted chunks are retrieved in response to a query or which terms in an index are accessed most frequently—can leak significant information about the underlying plaintext. Mitigating this, often through techniques like Oblivious RAM (ORAM), introduces yet more overhead. **Parameter Security** requires a delicate balancing act; cryptographers must choose parameters that are efficient enough for practical application performance while still providing a sufficient security level to protect data against both current and future threats, ensuring the long-term confidentiality of the encrypted bibliographic records.[3]

Chapter 4: Future Directions and Emerging Methods

4.1 Introduction to Future Directions

This chapter reframes the limitations outlined previously as a catalyst for innovation, charting a proactive research agenda to advance homomorphic encryption for bibliographic systems. The path forward is not merely about improving HE in isolation, but about orchestrating a multidisciplinary convergence of cryptography, systems engineering, and library science to bridge the gap between theoretical promise and practical deployment.[8]

4.2 Directions in Algorithmic and Methodological Innovation

Future breakthroughs will likely stem from hybrid architectures that strategically combine cryptographic tools. Hybrid Cryptographic Models that leverage the raw speed of Searchable Symmetric Encryption (SSE) for initial filtering with the computational power of FHE for private ranking or analysis on a reduced dataset offer a pragmatic balance of performance and functionality. Concurrently, HE-Aware Algorithm Design is crucial; developing new algorithms specifically engineered to minimize multiplicative depth and leverage SIMD parallelism natively can drastically reduce computational overhead. Furthermore, Machine Learning on Encrypted Metadata using the CKKS scheme presents a revolutionary direction, enabling privacy-preserving bibliographic analytics, such as training recommendation models or performing trend analysis directly on encrypted data, thus unlocking new value without compromising confidentiality.[16]

4.3 Directions in Performance Optimization

Addressing the performance bottleneck requires a multi-pronged hardware-software co-design approach. Hardware Acceleration through GPUs, FPGAs, and eventually ASICs dedicated to core HE operations like polynomial multiplication is essential for achieving the throughput necessary for large-scale systems. On the algorithmic side, research into Advanced Batching and Packing schemes will maximize computational density, ensuring every single homomorphic operation processes the maximum amount of data possible. Architecturally, Client-Aided Computation models can be explored, where clients with more trusted environments handle the most expensive operations like bootstrapping, thereby relieving the cloud server of this burden and creating a more feasible division of labor.[20]

4.4 Directions in System Integration and Usability

For HE to transition from research to practice, focus must shift to usability and integration. This includes community-driven Standardization Efforts for parameters, data formats, and APIs specific to information retrieval tasks, providing developers with clear and secure guidelines. The Development of Middleware is critical—this abstraction software would expose simple, high-level functions for “encrypted search” or “secure ranking,” hiding the underlying cryptographic complexity from bibliographic application developers. Finally, building and showcasing Use-Case Specific Prototypes for high-value tasks like encrypted cross-referencing or plagiarism detection will provide tangible proof of utility and stimulate wider adoption within the library community.[13-14]

5. Conclusion and Future Directions

Homomorphic encryption presents a promising path toward fully secure, privacy-preserving cloud-based bibliographic control systems. The ability to perform secure data chunking and encrypted metadata indexing directly on ciphertexts can mitigate the fundamental trust issues associated with cloud outsourcing. However, the journey from promise to practice is fraught with challenges. Performance remains the most significant hurdle, though continuous improvements in hardware acceleration (GPUs, FPGAs) and more efficient cryptographic schemes are steadily closing the gap. Secure and efficient content-aware chunking is a largely unsolved problem that requires

further research into optimized encrypted string processing. Furthermore, the integration of HE with existing bibliographic workflows and standards like BIBFRAME is an essential step for adoption, requiring collaboration between cryptographers and information scientists.

References

1. Bazm, M.-M., Lacoste, M., Sudholt, M., and Menaud, J.-M., Side-channels beyond the cloud edge: New isolation threats and solutions, 2017 1st Cyber Security in Networking Conference (CSNet), Rio de Janeiro, 2017, IEEE, 2017. https://doi.org/10.1109/csnet.2017.8241986
2. Lindemann, J. and Fischer, M., On the detection of applications in co-resident virtual machines via a memory deduplication side-channel, ACM SIGAPP Appl. Comput. Rev., 2019, vol. 18, no. 4, pp. 31–46. https://doi.org/10.1145/3307624.3307628
3. Hovhannisyan, H., Lu, K., Yang, R., Qi, W., Wang, J., and Wen, M., A novel deduplication-based covert channel in cloud storage service, 2015 IEEE Global Communications Conference (GLOBECOM), San Diego, Calif., 2015, IEEE, 2015, pp. 1–6. https://doi.org/10.1109/glocom.2015.7417228
4. Shin, Yo., Koo, D., and Hur, J., A survey of secure data deduplication schemes for cloud storage systems, ACM Comput. Surv., 2017, vol. 49, no. 4, pp. 1–38. https://doi.org/10.1145/3017428
5. Stanek, J. and Kencl, L., Enhanced secure thresholded data deduplication scheme for cloud storage, IEEE Trans. Dependable Secure Comput., 2016, vol. 15, no. 4, pp. 694–707. https://doi.org/10.1109/tdsc.2016.2603501
6. Bellare, M., Keelveedhi, S., and Ristenpart, T., Message-locked encryption and secure deduplication, Advances in Cryptology–EUROCRYPT 2013, Johansson, T. and Nguyen, P.Q., Eds., Lecture Notes in Computer Science, vol. 7881, Berlin: Springer, 2013, pp. 296–312. https://doi.org/10.1007/978-3-642-38348-9_18
7. Pooranian, Z., Chen, K.-Ch., Yu, Ch.-M., and Conti, M., RARE: Defeating side channels based on data-deduplication in cloud storage, IEEE INFOCOM 2018-IEEE Conf. on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, Hawaii, 2018, IEEE, 2018, pp. 444–449. https://doi.org/10.1109/infcomw.2018.8406888
8. Yu, Ch.-M., Gochhayat, S.P., Conti, M., and Lu, C., Privacy aware data deduplication for side channel in cloud storage, IEEE Trans. Cloud Comput., 2018, vol. 8, no. 2, pp. 597–609. https://doi.org/10.1109/tcc.2018.2794542
9. Saric, K., Ramachandran, G.S., Pal, S., Jurdak, R., and Nepal, S., A universal deduplication architecture for secure and efficient cloud storage, 2022 IEEE 4th Int. Conf. on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), Atlanta, 2022, IEEE, 2022, pp. 10–19. https://doi.org/10.1109/tps-isa56441.2022.00012
10. Tang, X., Liu, Z., Shao, Ya., and Di, H., Side channel attack resistant cross-user generalized deduplication for cloud storage, ICC 2022–IEEE Int. Conf. on Communications, Seoul, Republic of Korea, 2022, IEEE, 2022, pp. 998–1003. https://doi.org/10.1109/icc45855.2022.9838727

11. Koo, D. and Hur, J., Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing, Future Gener. Comput. Syst., 2018, vol. 78, pp. 739–752. https://doi.org/10.1016/j.future.2017.01.024
12. Huang, D., Zhou, J., Mi, B., Kuang, F., and Liu, Ya., Key-based data deduplication via homomorphic NTRU for internet of vehicles, IEEE Trans. Veh. Technol., 2022, vol. 72, no. 1, pp. 239–252. https://doi.org/10.1109/tvt.2022.3205627
13. Dave, J., Dutta, A., Faruki, P., Laxmi, V., and Gaur, M.S., Secure proof of ownership using merkle tree for deduplicated storage, Autom. Control Comput. Sci., 2020, vol. 54, no. 4, pp. 358–370. https://doi.org/10.3103/S0146411620040033
14. Dave, J., Faruki, P., Laxmi, V., Bezawada, B., and Gaur, M., Secure and efficient proof of ownership for deduplicated cloud storage, Proc. 10th Int. Conf. on Security of Information and Networks, Jaipur, India, 2017, New York: Association for Computing Machinery, 2017, pp. 19–26. https://doi.org/10.1145/3136825.3136889
15. Dave, J., Faruki, P., Laxmi, V., Zemmari, A., Gaur, M., and Conti, M., SPARK: Secure pseudorandom key-based encryption for deduplicated storage, Comput. Commun., 2020, vol. 154, pp. 148–159. https://doi.org/10.1016/j.comcom.2020.02.037
16. Dave, J., Saharan, S., Faruki, P., Laxmi, V., and Gaur, M.S., Secure random encryption for deduplicated storage, Information Systems Security, Shyamasundar, R., Singh, V., and Vaidya, J., Eds., Lecture Notes in Computer Science, vol. 10717, Cham: Springer, 2017, pp. 164–176. https://doi.org/10.1007/978-3-319-72598-7_10
17. Taranin, S.M., Deduplication in the backup system with information storage in a database, Autom. Control Comput. Sci., 2018, vol. 52, no. 7, pp. 608–614. https://doi.org/10.3103/s0146411618070246
18. Alguliyev, R.M., Imamverdiyev, Y.N., and Abdullayeva, F.J., PSO-based load balancing method in cloud computing, Autom. Control Comput. Sci., 2019, vol. 53, no. 1, pp. 45–55. https://doi.org/10.3103/s0146411619010024
19. Xue, S. and Ren, C., Security protection of system sharing data with improved CP-ABE encryption algorithm under cloud computing environment, Autom. Control Comput. Sci., 2019, vol. 53, no. 4, pp. 342–350. https://doi.org/10.3103/S0146411619040114
20. Kumar, P., Gupta, G.P., and Tripathi, R., Design of anomaly-based intrusion detection system using fog computing for IoT network, Autom. Control Comput. Sci., 2021, vol. 55, no. 2, pp. 137–147. https://doi.org/10.3103/S0146411621020085
21. Jay Dave, Nikumani Choudhury Secure and Efficient Traffic Obfuscation Scheme for Deduplicated Cloud Storage. Aut. Control Comp. Sci. 58, 153–165 (2024).
22. Ruba, S., Kalpana, A.M. Advanced chunk-based data deduplication framework for secure data storage in cloud using hybrid heuristic assisted optimal key-based encryption. Wireless Netw 31, 3467–3489 (2025).