Educational Administration: Theory and Practice
2024, 30(5), 9421-9429
ISSN: 2148-2403
https://kuey.net/

Research Article

Securing Cloud Memory Through Efficient Deduplication Using Ecc Algorithm

Jibin Joy[1*], Dr. S. Devaraju[2]

[1*]Research Scholar (Ph.D.), Department of Computer Science, Sri Krishna Arts and Science College,Coimbatore, Tamil Nadu, India. jibinjoysamuel@gmail.com
[2]Senior Assistant Professor, VIT Bhopal University, Bhopal, Madhya Pradesh, India. devamcet@gmail.com

Citation: Jibin Joy, Dr. S. Devaraju, (2024), Securing Cloud Memory Through Efficient Deduplication Using Ecc Algorithm, Educational Administration: Theory and Practice, 30(5), 9421-9429
DOI: 10.53555/kuey.v30i5.4583

ARTICLE INFO

ABSTRACT

client-side data deduplication framework designed for enterprise utilization in cloud storage environments allows enterprises to conduct internal cross-user data deduplication and subsequently outsource the deduplicated data to the cloud. Furthermore, the cloud service provider performs cross-enterprise data deduplication to eliminate duplicates, leading to substantial cost and space savings. The framework's primary goal is to empower enterprises to securely and efficiently execute various operations, such as searching through encrypted data, sharing data within the organization, and directly downloading data from the cloud, all without needing to place complete trust in the cloud service provider. The study delves into cloud data security, focusing on a unique approach that merges reversible polynomial hashing with an Enhanced ECC (Elliptic Curve Cryptography) algorithm. The objective is to assess the efficacy of this hybrid cryptographic method in securing cloud-stored data. The research aims to explore how reversible polynomial hashing enhances security and how its integration with Enhanced ECC boosts data protection. This phase entails a thorough literature review to lay the groundwork, followed by designing and implementing the proposed algorithm. The methodology includes encryption, storage, and potential decryption procedures. Through a methodical evaluation of the algorithm's strength and performance, the study aims to offer valuable insights into leveraging innovative cryptographic techniques for improved cloud data security.

Keywords: Information Management Table (IMT), Elliptic Curve Cryptography(ECC), Fuzzy Identity-Based Encryption (FIBE), media access control (MAC)

1. INTRODUCTION:

The rapid expansion of cloud computing has made data storage, access, and security practices. With organizations increasingly relying on cloud infrastructure for managing digital assets, there is a critical demand for advanced methodologies to improve memory management, enhance data security, facilitate efficient data sharing, and ensure robust integrity verification. This extensive research is structured into distinct phases, each addressing key aspects of cloud computing. The initial phase focuses on enhancing cloud memory management by employing flat block size deduplication techniques. By addressing common challenges and assessing the impact of deduplication methods tailored for flat block sizes, this phase aims to provide insights for optimizing performance in cloud environments. A thorough literature review sets the groundwork, followed by practical implementation and testing to evaluate essential performance metrics. Moving to data security, the second phase introduces a new approach that combines reversible polynomial hashing with an Enhanced ECC algorithm. The initial main objective is to evaluate the effectiveness of this hybrid cryptographic scheme in strengthening data protection in cloud storage. Through careful design, implementation, and evaluation, this phase aims to showcase how innovative cryptographic techniques can enhance security measures in cloud environments. Furthermore, in cloud data centers, the practice of over-provisioning servers to accommodate peak request demands results in significant energy wastage. This innovative approach to cloud computing and data management aims to optimize resource utilization, enhance data security, and minimize energy waste in cloud data centers. The paper delves into real-world applications and case studies demonstrating the successful implementation of flat block size deduplication. These instances showcase the potential of this technique to enhance scalability, reduce costs, and improve overall system reliability in cloud environments. This

comprehensive exploration sets various stages for understanding the relevance and significance of flat block size deduplication techniques in modern computing infrastructures. [21] Fig 1.1 gives the brief description regarding the benefits of cloud computing in the real world architecture.

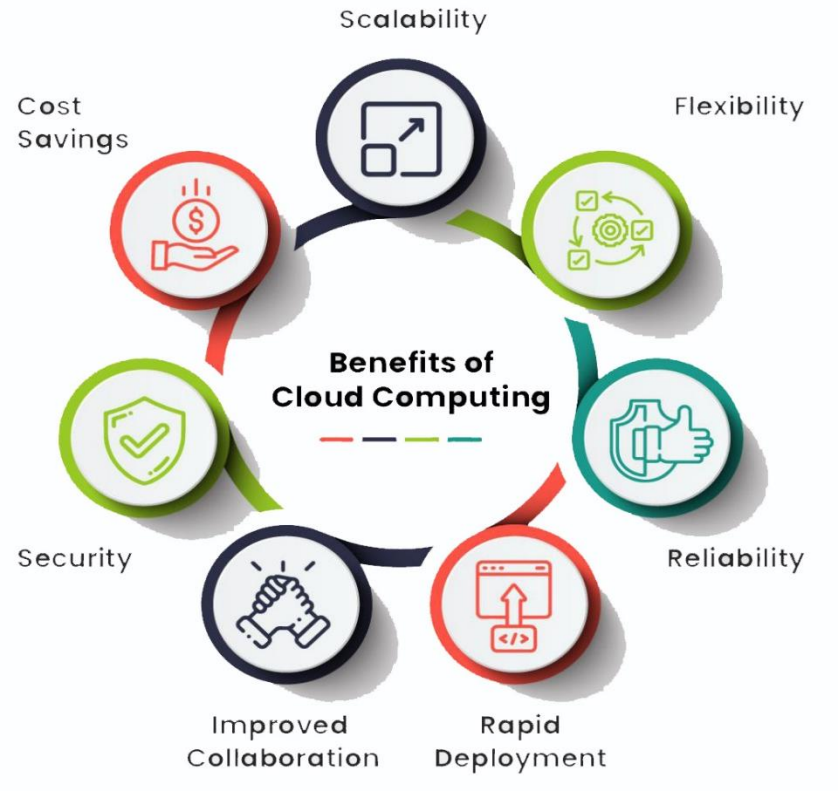

Fig:1.1 Benefits of Cloud Computing

2. NEED FOR STUDY:

In their study, Bosman et al. [1] drew attention to the risks associated with memory deduplication, highlighting how seemingly harmless features could be exploited by sophisticated attackers. They demonstrated that primitives based on deduplication could be used to expose sensitive information, posing a threat to system security. This underscores the relevancy of implementing high level security measures in deduplication strategies. Cui et al. [3] introduced UWare, a middleware aimed at enhancing data transfer efficiency through encrypted cloud storage. Their research aimed to address concerns about side-channel leakage while retaining the benefits of deduplication. They focused on leveraging similarity features and employing the Proof-of-Work (PoW) protocol to balance deduplication efficiency with system throughput. Garg et al. [5] utilized graphics processing units to enhance deduplication efficiency. Their proposed method, Catalyst, offloaded the memory deduplication workload to GPU devices, enabling rapid identification of potentially duplicated pages. Results indicated that Catalyst significantly expedited data sharing from memory compared to traditional approaches. Kaur et al. [9] explored data deduplication in cloud computing to reduce storage costs, network traffic, and power consumption. They stressed the importance of innovative deduplication methods to enhance the efficiency of large-scale storage systems. Ning et al. [11] introduced group-based memory deduplication as a novel defense against covert channel attacks in multi-tenant clouds. Their approach provided group-level isolation, safeguarding virtual machines (VMs) from side-channel attacks by enabling same-group members to manage guest memory using shared secrets. Raoufi et al. [13] presented PageCmp, a memory-based page comparator that minimized data transmission during deduplication. By leveraging charge-sharing phenomena in DRAM's bulk bitwise operations, they significantly reduced bandwidth usage while maintaining acceptable execution time and power consumption levels. Vano-Garcia and Marco-Gisbert [15] examined how kernel randomization impacts hypervisors' ability to optimize memory deduplication. Their study highlighted the memory overhead introduced by kernel randomization, illustrating the challenges of integrating cloud computing with security solutions. Block size deduplication is a method employed in storage systems to enhance storage efficiency by identifying and removing duplicate data blocks [16]. This technique involves segmenting data into fixed-size blocks and identifying identical blocks within the dataset. Instead of storing redundant blocks multiple times, block size deduplication retains only one copy of each unique block and uses references to that block for duplicates. By eliminating redundant data, this approach reduces storage overhead and enhances storage efficiency, particularly in environments with multiple copies of the same data. Block size deduplication is commonly utilized in backup solutions, file systems, and storage devices to optimize storage space and improve data management. Boundary deduplication is a data optimization method that targets and

removes duplicate data within specific boundaries or segments like files or logical units, rather than at the block level. The process involves identifying these boundaries, breaking down the data into smaller parts, and comparing them using hash functions. By matching hash values, duplicate parts are pinpointed, enabling the system to store just one copy of each unique part and substitute duplicates with references. This technique saves storage space, boosts data retrieval speed, and enhances storage efficiency, proving valuable in backup systems, cloud storage, and data archiving to streamline data management and storage utilization.

3. PROPOSED METHODOLOGY:

The study explores cloud data security, focusing on a new approach that combines reversible polynomial hashing with an Enhanced ECC (Elliptic Curve Cryptography) algorithm. The aim is to examine the effectiveness of this hybrid cryptographic method in protecting cloud-stored data. The research aims to investigate how reversible polynomial hashing improves security and how its integration with Enhanced ECC enhances data protection. This phase includes an in-depth literature review to establish a base, followed by designing and implementing the proposed algorithm. The methodology covers encryption, storage, and potential decryption procedures. Through a methodical assessment of the algorithm's strength and performance, the study endeavors to offer practical insights into utilizing innovative cryptographic techniques for improved cloud data security. Figure 3.1 illustrates the proposed work flow architecture of the e ECC (Elliptic Curve Cryptography) algorithm.

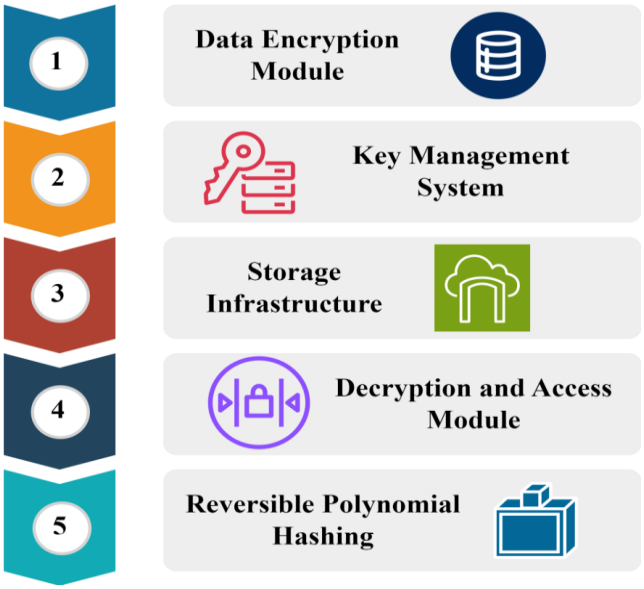

Fig 3.1 Proposed workflow architecture

4. IMPLEMENTATION

4.1.1 Polynomial Hashing

When each variable x_i in the polynomials has a degree of n, there must be a minimum of n leaves labeled by x_i in the formula, as mentioned earlier. This implies that any algebraic formula computing the polynomial qn i=1 xn i must have at least n^2 leaves. Consequently, there should be a minimum of n leaves, where n^2 is required. Hence, significantly more than the sum of the individual variable degrees is needed for meaningful and non-trivial lower bounds for algebraic formulas. In cases where the target polynomial has a constant individual degree (essentially 1 without loss of generality), there exists an intriguing parameter regime.

The first non-trivial lower bound for algebraic formulas was established for an n-variate multilinear polynomial, specifically the Ôn ⋄ Ôn determinant, which was shown to be (n^(3/2)). Nec66 demonstrated that the size of a Boolean formula in the complete binary basis has a lower bound of (n^2/log n). This study can prove a lower bound of (n^2/log n) for an alternative multilinear polynomial using arguments similar to those in: consider two sets of n variables each, Ó x(j) i: i œ [log n],j œ [n/ log n] Ô and {yk: k œ [n]}, and choose a bijection ÿ: 2[log n] ae [n]. It will be the difficult multilinear polynomial.

$$\sum_{j=1}^{n/\log n} \left(\sum_{SC[\log n]} [n] \right) [y_i(s)] . \prod_{i \in S} x_i^{(j)} \quad \text{---------- (5.1)}$$

However, the Nechipurok and Kalorkoti method fails to demonstrate a lower bound that is asymptotically superior to "n^2/log n".

This study revises the concepts from equation 5.1 in Theorem 5.1 to establish a lower limit of (n^2) for a specific multilinear polynomial with n variables.

The n-variate elementary symmetric polynomial of degree 0.1n, which can be computed by a formula (specifically a depth-3 formula) of size O(n^2) over all sufficiently large fields, as observed by Ben-Or. A lower

bound of (n log n) on the more stringent concept of circuit complexity is the sole known super-linear lower bound on the general formula complexity of basic symmetric polynomials before this research. Based on the findings of this study, it appears that Kalorkoti's evidence does not enhance the formula complexity of these polynomials. Therefore, the lower limit proposed by this research suggests a super-linear separation in the formula complexity and circuit complexity of a specific family of multilinear polynomials. Hrubeš et al. introduced the concept of ordered polynomials in their work. Ordering a homogeneous polynomial of degree d involves ensuring that for each monomial, variables appearing at position k can only come from the k-th bucket. This is achieved by dividing the set of variables it depends on into d buckets. This study expands on this concept by making the bucket indices location-independent. In other words, it is not necessary for a variable at position k to originate from the k-th bucket; what matters is that in each monomial, the variables must appear in a non-decreasing sequence of their bucket indices. Analogous to an abecedarian word, which is a word with letters arranged alphabetically, these polynomials are referred to as abecedarian in this research.

The distinction between ordered polynomials and abecedarian ones can be succinctly explained using regular expressions from Automata Theory. In the context of a non-commutative polynomial f in F Æx1,...,xnÍ, where the variables are divided into buckets {X_1,...,X_m}, a monomial in f is considered ordered with respect to {X_1,...,X_m} if it can be generated using the regular expression X_1 ••• X_m. On the other hand, F is classified as abecedarian if its monomials are words that can be constructed using the regular expression X_ú 1 ••• X_ú m.

It is evident that any ordered polynomial is also abecedarian with respect to the same partition, due to the increasing position indices. Consider the following variant of the complete homogeneous symmetric polynomial as an example:

〚CHSYM〛 *(n,d)^((ord)) (x)*=∑(1≤i1≤⋯≤i_d≤n)▒〚x_(i_1)^((1))...x_(i_d)^((d)) 〛 ---------- (5.2)

However, abecedarian properties are not essential for the naturally occurring non-commutative versions of commutative computation. Therefore, not all non-commutative representations of commutative polynomials are abecedarian with respect to the set {X_i : X_i = {x_i }}. For instance, consider the arc-full rank polynomial constructed by Dvir, Malod, Perifel, and Yehudayoff to demonstrate a super-polynomial separation of formula and ABP powers in a multilinear context. This polynomial, denoted as f, can be examined as f', a non-commutative polynomial, in this research context. This chapter introduces a system based on the sponge building principle. The initial register has a capacity of c bits and a rate of r bits, with the state b described by the equation b = r + c. The state b consists of 256 bits, where r=32 and c=224, representing a 32-bit rate and a 224-bit capacity, respectively.

The message block is written into the most significant 32 bits of the initial register, following the sponge building method. The message is padded to split it into 32-bit chunks. The first message block is XORed with the initial register, changing the initial state s0 to all zeroes, i.e., s0 = 0r || 0c. In this updated register, sixteen 16-bit words are identified. To introduce confusion, the PRESENT S-box (4 × 4) is simultaneously applied to these sixteen words. Words one through three are then modified by XORing with word four, while word four remains unchanged. To achieve word-wise diffusion, it is necessary to apply this imbalanced Feistel structure in parallel on four 64-bit strings of the register. Round constants are added to the 256-bit register after rotating it 8 bits to the left to ensure general diffusion. The integers are added modulo 2^16 to maintain consistency.

A single round of updates is applied to the registry, incorporating the S-Box layer for confusion, XORing and rotation for general diffusion, and modular addition to eliminate established patterns. This process is repeated 31 more times for a total of 32 rounds, each applied to every message block.

Algorithm

Input:

Consider a set of variables x_1, x_2, ..., x_n, where n represents the number of variables in the polynomial.

Steps:

1. Select a Polynomial: Choose a polynomial equation to hash, which can be a multilinear polynomial or any other type of polynomial based on your specific needs.
2. Calculate Polynomial Hash: Begin by setting an accumulator variable to zero.
3. For each term in the polynomial: Calculate the hash value for each term using the variable values in that term.
4. Utilize a hash function such as SHA-256 or MD5 for this calculation.
5. Add the computed hash value to the accumulator variable.
6. Once all terms have been processed, the accumulator variable will hold the hash value for the entire polynomial.

Output:

The hash value derived from the accumulator variable serves as the polynomial hash.

4.1.2 Polynomial hashing with Enhanced ECC

Polynomial hashing with Enhanced Elliptic Curve Cryptography (EECC) merges the concepts of polynomial hashing, where varying input lengths are converted into fixed-length hash values using polynomial functions, with the advanced security features of EECC that operate on elliptic curves across finite fields. This combined method involves subjecting the input message to polynomial hashing to produce a hash value, which is then incorporated into EECC processes like key generation, encryption, and decryption. By uniting the strengths of polynomial hashing's mathematical properties and the robustness of elliptic curve cryptography, this fusion improves the security and efficiency of cryptographic operations. The resulting hash value from polynomial hashing plays a pivotal role in cryptographic protocols, ensuring data integrity, authentication, and confidentiality. Optimal parameter selection and meticulous implementation are vital to maximize the efficacy and resilience of this integrated approach in safeguarding digital communications and data transfers.

Encryption Process:

- Each block in the color component data undergoes encryption individually.
- The encryption algorithm operates on segments within each block.
- The blocks are identified by their row and column indices, denoted as b(i, j).
- Two halves of the input data are utilized in the encryption process, namely D_x(i, j) and D_y(i+1, j).
- Encryption steps:

1. Calculate C_1 = H * P_e.
2. Determine C_2 = (D_x, D_y) + C_1.

Decryption Process:

- The decryption process utilizes the private key (H) and the point C11 for decoding.
- Decryption steps:

1. Compute C_11 = H * C_1.
2. Obtain C_ij = C_2 - C_11, representing the decrypted result.

Finalization:

- Extract pixel values from C_ij to retrieve the original color components (R, G, B) separately.
- Combine the separate color components to reconstruct the decrypted image data as F_image = ∑(RGB).

Algorithm:

Input:

- Message (Message), Public Key (P_e), Private Key (H), Color Component Data

Steps:

1. Split the data into blocks denoted as b(i, j), where i and j represent row and column indices.
2. For each block (i, j):
   - Extract data sets D_x(i, j) and D_y(i+1, j) from the block.
   - Calculate C1 = H * P_e.
   - Compute C2 = (D_x, D_y) + C1.
3. Generate Enc_Data containing all encrypted blocks.
4. For each block in Enc_Data:
   - Compute C11 = H * C1.
   - Determine C_ij = C2 - C11 to derive the decrypted block.
5. Retrieve pixel values from C_ij and reconstruct the original color components (R, G, B) for each pixel.
6. Combine pixel components to produce Dec_data as the decrypted data.

Output:

- Encrypted data (Enc_data)

5. RESULT ANALYSIS:

Performance evaluation is vital for assessing the effectiveness and efficiency of systems, processes, or technologies. It includes measuring and analyzing different metrics and parameters to gauge how well a system or component meets its objectives. In the realm of cloud memory management with flat block size deduplication techniques, performance evaluation is pivotal. Through thorough assessment of critical metrics like storage space savings, access times, and computational overhead, researchers can glean valuable insights into how deduplication methods impact efficiency and resource utilization in cloud settings.

Table 5.1: Encryption time comparison table

File Size (KB)	**Encryption Time**		
	ECC	polynomial hashing	polynomial hashing with Enhanced ECC
1	0.11	0.9	0.7
2	0.27	0.43	1.21
4	1.61	1.32	1.11
8	2.24	1.7	1.2

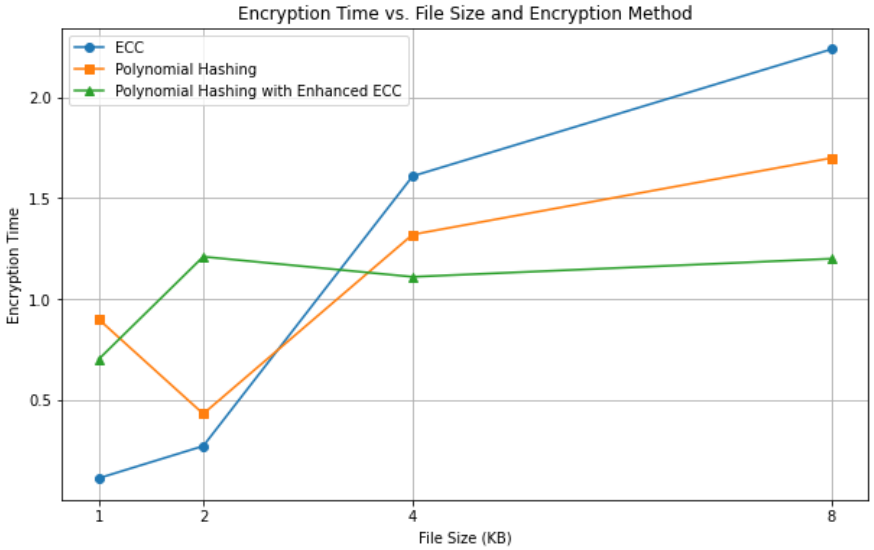

Figure 5.1: Encryption time comparison chart

In Figure 5.1 and Table 5.1, the encryption times (in seconds) for different file sizes (in kilobytes) were compared across three encryption methods: ECC, polynomial hashing, and polynomial hashing with Enhanced ECC. For a 1KB file, ECC took 0.11 seconds, polynomial hashing took 0.9 seconds, and polynomial hashing with Enhanced ECC took 0.7 seconds.

- As file sizes increased to 2KB, 4KB, and 8KB, encryption times rose for all methods. Polynomial hashing consistently exhibited shorter encryption times compared to ECC and polynomial hashing with Enhanced ECC, especially noticeable with larger file sizes. Polynomial hashing with Enhanced ECC showed slightly longer encryption times than polynomial hashing alone but remained faster than ECC. These findings indicate that polynomial hashing, especially when combined with Enhanced ECC, can provide efficient encryption across different file sizes, potentially striking a balance between security and performance in data encryption applications.

Table 5.2: Decryption time comparison table

File Size (KB)	**Decryption Time**		
	ECC	polynomial hashing	polynomial hashing with Enhanced ECC
1	0.1351	0.0303	0.0248
2	0.0498	0.0388	0.0214
4	0.0614	0.0523	0.0329
8	0.0943	0.0824	0.0565

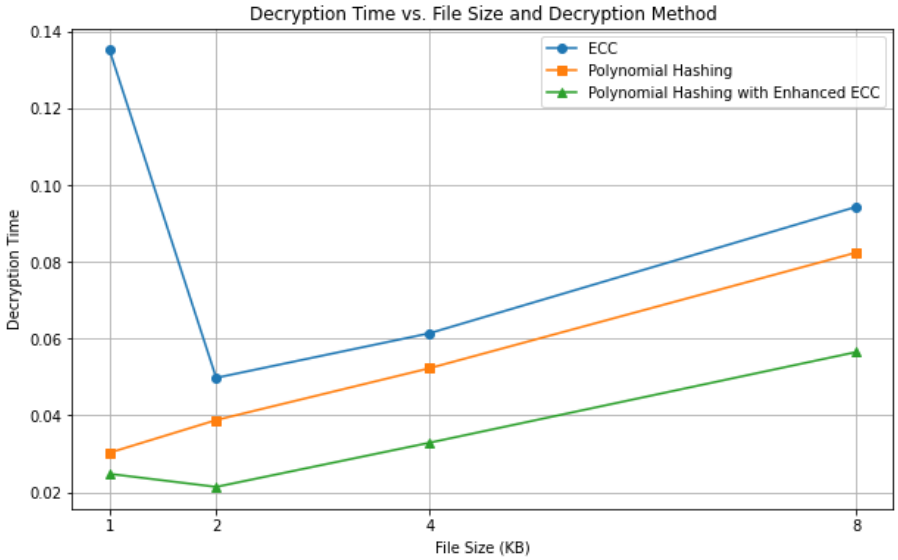

Figure 5.2: Decryption time comparison chart

In Figure 5.2 and Table 5.2, the decryption times (in seconds) for different file sizes (in kilobytes) were analyzed across three decryption methods: ECC, polynomial hashing, and polynomial hashing with Enhanced ECC. For a 1KB file, ECC took 0.1351 seconds, polynomial hashing took 0.0303 seconds, and polynomial hashing with Enhanced ECC took 0.0248 seconds. As file sizes increased to 2KB, 4KB, and 8KB, decryption times increased for all methods.

- Polynomial hashing consistently demonstrated shorter decryption times compared to ECC and polynomial hashing with Enhanced ECC, especially notable with larger file sizes. Polynomial hashing with Enhanced ECC exhibited slightly longer decryption times than polynomial hashing alone but remained faster than ECC. These results suggest that polynomial hashing, especially when enhanced with ECC, can deliver efficient decryption performance across various file sizes, potentially striking a balance between security and decryption speed in data decryption applications.

Table 5.3: Downloading time comparison table

File Size (KB)	**Downloading time**		
	ECC	polynomial hashing	polynomial hashing with Enhanced ECC
1	0.9	0.7	0.5
2	0.38	0.57	1.38
4	1.71	1.81	1.01
8	2.24	1.6	0.95

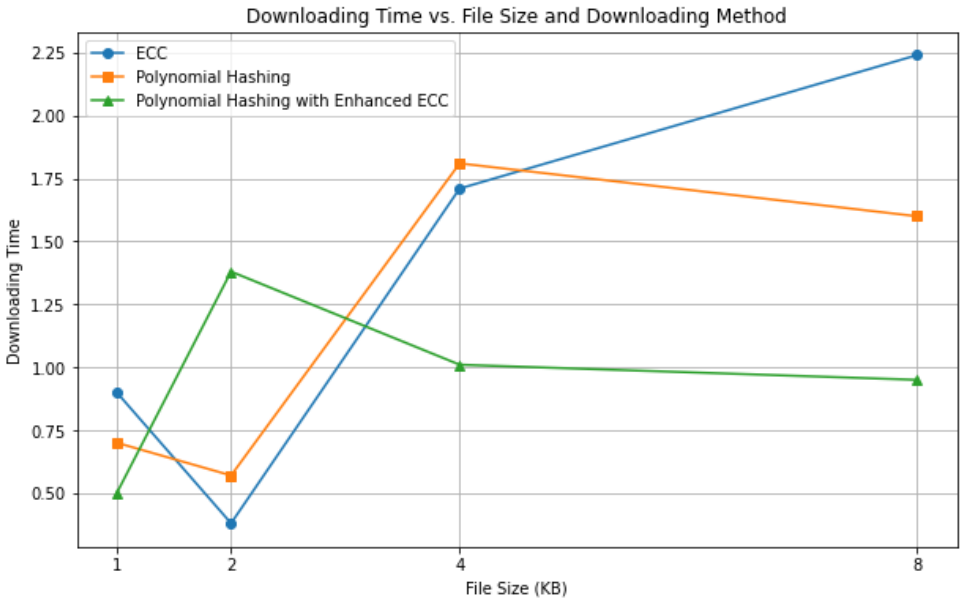

Figure 5.3: Downloading time comparison chart

In Figure 5.3 and Table 5.3, the provided data details the download times (in seconds) for various file sizes (in kilobytes) using three different download methods: ECC, polynomial hashing, and polynomial hashing with Enhanced ECC. For a 1KB file, download times were 0.9 seconds for ECC, 0.7 seconds for polynomial hashing, and 0.5 seconds for polynomial hashing with Enhanced ECC. As file sizes increased to 2KB, 4KB, and 8KB, download times generally increased across all methods. Polynomial hashing consistently showed shorter download times compared to ECC and polynomial hashing with Enhanced ECC, particularly noticeable with larger file sizes. Polynomial hashing with Enhanced ECC had slightly higher download times than polynomial hashing alone but remained lower than ECC. This data indicates that polynomial hashing, especially when enhanced with ECC, can deliver efficient download performance for different file sizes, potentially offering a balance between data security and download speed in cloud storage applications.

Table 5.4: Data Validation Accuracy comparison table

File Size (KB)	**Data Validation Accuracy**		
	ECC	polynomial hashing	polynomial hashing with Enhanced ECC
10	0.88	0.89	0.93
40	0.85	0.89	0.94
80	0.89	0.94	0.95
100	0.93	0.94	0.97

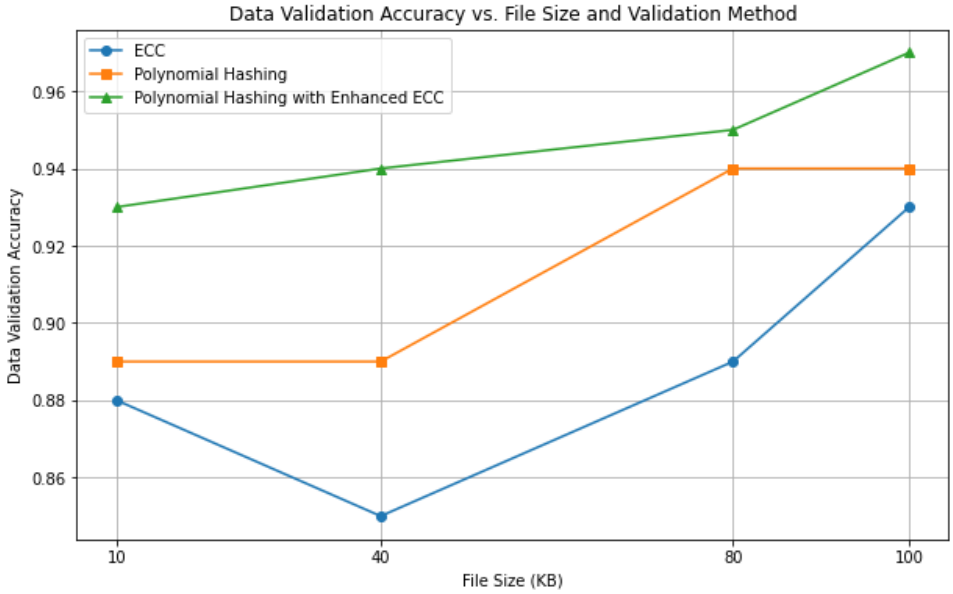

Figure 5.4: data validation accuracy comparison chart

In Figure 5.4 and Table 5.4, the data presents data validation accuracy, measured in decimal values, for different file sizes (in kilobytes) using three methods: ECC, polynomial hashing, and polynomial hashing with Enhanced ECC. Across file sizes of 10KB, 40KB, 80KB, and 100KB, polynomial hashing with Enhanced ECC consistently showed the highest data validation accuracy, ranging from 0.93 to 0.97. Polynomial hashing alone also exhibited high accuracy, slightly below Enhanced ECC, with values ranging from 0.89 to 0.94. ECC, although still providing acceptable accuracy, had lower values compared to the other methods, with accuracies ranging from 0.85 to 0.93. These findings suggest that both polynomial hashing and polynomial hashing with Enhanced ECC are effective for data validation, with Enhanced ECC demonstrating slightly higher accuracy across different file sizes.

6. CONCLUSION AND FUTURE WORK

The study presented in this paper concentrates on improving cloud data security by utilizing a hybrid cryptographic technique that merges reversible polynomial hashing with an Enhanced ECC algorithm. The primary goal of the research is to assess the efficacy of this method in safeguarding data stored in the cloud by examining its security enhancements and integration advantages. The research paper not only addresses current challenges and solutions in cloud computing but also sets the stage for future advancements. Future research could explore advanced deduplication techniques for various data types, implement quantum-resistant cryptographic protocols for improved data security, develop dynamic access control mechanisms for adaptable data sharing, integrate blockchain technology for comprehensive data governance, utilize AI-driven optimization and automation for streamlined cloud operations, consider ethical and regulatory aspects of cloud computing, and focus on hybrid and multi-cloud architectures for better interoperability and resource

utilization. These research directions aim to optimize cloud infrastructures, enhance security measures, improve data governance, and ensure responsible cloud computing practices in the future.

7. REFERENCES

1. Bosman, E., Razavi, K., Bos, H., & Giuffrida, C. (2016). Dedup Est Machina: Memory Deduplication as an Advanced Exploitation Vector. 2016 IEEE Symposium on Security and Privacy (SP). doi:10.1109/sp.2016.63
2. Cui, H., Duan, H., Qin, Z., Wang, C., & Zhou, Y. (2019). SPEED: Accelerating Enclave Applications Via Secure Deduplication. 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS). doi:10.1109/icdcs.2019.00110
3. Cui, H., Wang, C., Hua, Y., Du, Y., & Yuan, X. (2018). A Bandwidth-Efficient Middleware for Encrypted Deduplication. 2018 IEEE Conference on Dependable and Secure Computing (DSC). doi:10.1109/desec.2018.8625127
4. Fu, Y., Xiao, N., Jiang, H., Hu, G., & Chen, W. (2017). Application-Aware Big Data Deduplication in Cloud Environment. IEEE Transactions on Cloud Computing, 1–1. doi:10.1109/tcc.2017.2710043
5. Garg, A., Mishra, D., & Kulkarni, P. (2017). Catalyst. Proceedings of the 13th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments - VEE '17. doi:10.1145/3050748.3050760
6. Huang, H., Yan, C., Liu, B., & Chen, L. (2017). A survey of memory deduplication approaches for intelligent urban computing. Machine Vision and Applications, 28(7), 705–714. doi:10.1007/s00138-017-0834-6
7. Jagadeeswari, N., & Mohanraj, V. (2017). A survey on memory deduplication employed in cloud computing. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS). doi:10.1109/icecds.2017.8390074
8. Jia, S., Wu, C., & Li, J. (2017). Loc-K: A Spatial Locality-Based Memory Deduplication Scheme with Prediction on K-Step Locations. 2017 IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS). doi:10.1109/icpads.2017.00049
9. Kaur, R., Chana, I., & Bhattacharya, J. (2017). Data deduplication techniques for efficient cloud storage management: a systematic review. The Journal of Supercomputing, 74(5), 2035–2085. doi:10.1007/s11227-017-2210-8
10. Kim, D., Song, S., & Choi, B.-Y. (2016). Existing Deduplication Techniques. Data Deduplication for Data Optimization for Storage and Network Systems, 23–76. doi:10.1007/978-3-319-42280-0_2
11. Ning, F., Zhu, M., You, R., Shi, G., & Meng, D. (2016). Group-Based Memory Deduplication against Covert Channel Attacks in Virtualized Environments. 2016 IEEE Trustcom/BigDataSE/ISPA. doi:10.1109/trustcom.2016.0063
12. Niu, Y., Liu, W., Xiang, F., & Wang, L. (2015). Fast Memory Deduplication of Disk Cache Pages in Virtual Environments. 2015 IEEE Fifth International Conference on Big Data and Cloud Computing. doi:10.1109/bdcloud.2015.50
13. Raoufi, M., Deng, Q., Zhang, Y., & Yang, J. (2019). PageCmp: Bandwidth Efficient Page Deduplication through In-memory Page Comparison. 2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI). doi:10.1109/isvlsi.2019.00023
14. Saharan, S., Somani, G., Gupta, G., Verma, R., Gaur, M. S., & Buyya, R. (2020). QuickDedup: Efficient VM deduplication in cloud computing environments. Journal of Parallel and Distributed Computing. doi:10.1016/j.jpdc.2020.01.002
15. Vano-Garcia, F., & Marco-Gisbert, H. (2018). How Kernel Randomization is Canceling Memory Deduplication in Cloud Computing Systems. 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA). doi:10.1109/nca.2018.8548338
16. Veni, T., & Bhanu, S. M. S. (2014). MDedup++: Exploiting Temporal and Spatial Page-Sharing Behaviors for Memory Deduplication Enhancement. The Computer Journal, 59(3), 353–370. doi:10.1093/comjnl/bxu149
17. Wang, C., Wei, Q., Yang, J., Chen, C., Yang, Y., & Xue, M. (2018). NV-Dedup: High-Performance Inline Deduplication for Non-Volatile Memory. IEEE Transactions on Computers, 67(5), 658–671. doi:10.1109/tc.2017.2774270
18. Wu, J., Hua, Y., Zuo, P., & Sun, Y. (2018). Improving Restore Performance in Deduplication Systems via a Cost-efficient Rewriting Scheme. IEEE Transactions on Parallel and Distributed Systems, 1–1. doi:10.1109/tpds.2018.2852642
19. Xia, W., Jiang, H., Feng, D., Douglis, F., Shilane, P., Hua, Y., … Zhou, Y. (2016). A Comprehensive Study of the Past, Present, and Future of Data Deduplication. Proceedings of the IEEE, 104(9), 1681–1710. doi:10.1109/jproc.2016.2571298
20. Zuo, P., Hua, Y., Zhao, M., Zhou, W., & Guo, Y. (2018). Improving the Performance and Endurance of Encrypted Non-Volatile Main Memory through Deduplicating Writes. 2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). doi:10.1109/micro.2018.00043.
21. https://www.infosectrain.com/blog/real-world-applications-of-cloud-computing/