OPTIMIZING CLOUD BASED DATA STORAGE WITH POLYNOMIAL HASHING AND BUFFERING TECHNIQUES

Sowndharya M[1*], Duraisamy S[2], Pavithra M[3]
[1]Research Scholar, Chikkanna Government Arts College, Tiruppur, Tamil Nadu, India
[2] Research Supervisor, Chikkanna Government Arts College, Tiruppur, Tamil Nadu, India
[3] Research Scholar, Chikkanna Government Arts College, Tiruppur, Tamil Nadu, India
* Corresponding author: sowndharyacgac2022@gmail.com

ABSTRACT

In the ever-evolving landscape of data management, this paper focuses on the secure storage of groundwater level data in the cloud. Leveraging the capabilities of cloud-based databases like Amazon Web Services (AWS) and Microsoft Azure, this phase aims to establish a robust framework that ensures high security, accessibility, and scalability of the stored data. Storing groundwater level data in the cloud offers a myriad of advantages. It guarantees easy and efficient access to the data from any corner of the globe, enabling real-time analysis and visualization. Additionally, cloud storage provides a secure backup mechanism, safeguarding the data against loss or corruption incidents. This abstract highlights the implementation of a cutting-edge buffering polynomial hashing algorithm, designed to enhance the security of the stored data, ensuring its integrity and confidentiality. The utilization of cloud-based solutions not only simplifies data accessibility but also empowers researchers and stakeholders with real-time insights into groundwater level patterns and trends. Through the integration of advanced data analytics tools, the stored data can be meticulously analyzed, unveiling valuable insights that are instrumental in decision-making processes related to water resource management and environmental conservation.

Keywords: *Amazon Web Services, Cloud Storage, Cloud-Based Databases, High Security, Secure Storage*

INTRODUCTION

In our increasingly digitized world, the management and analysis of vast datasets have become pivotal for scientific research, environmental monitoring, and decision-making processes [1-2]. One of the critical domains where data holds immense importance is in understanding groundwater levels. The availability and quality of groundwater significantly impact agriculture, ecosystems, and human populations [3-4]. Consequently, efficient and secure methods of collecting, storing, and analyzing groundwater data are essential for sustainable water resource management [5-6]. This study enters with a focus on elevating the standards of data storage, accessibility, and security in the realm of groundwater level monitoring [7-8]. Harnessing the power of cloud-based technologies, specifically employing platforms like Amazon Web Services (AWS) or Microsoft Azure, this phase endeavors to revolutionize the way groundwater data is handled [9-10]. The shift to cloud storage promises unparalleled advantages, including enhanced accessibility, real-time analysis, visualization capabilities, and robust backup solutions [11].

This introduction sets the stage for exploring the multifaceted benefits of cloud-based storage solutions, emphasizing their role in ensuring high security, scalability, and accessibility for groundwater level data [12-13]. Additionally, it delves into the pivotal role of real-time data analysis and visualization in unveiling patterns and trends, providing valuable insights for scientific research and policy formulation [14-15]. Moreover, this phase introduces a sophisticated buffering polynomial hashing algorithm, a cutting-edge encryption technique employed to fortify the security of stored data, ensuring its integrity and confidentiality in an era where data breaches pose significant risks [16-17]. By amalgamating the power of cloud technology with advanced encryption methods, this study aims to not only address the challenges of secure data storage but also empower researchers and policymakers with tools to make informed decisions about water resource management [18-19]. The subsequent sections will delve deeper into the methodologies employed, the benefits accrued, and the insights gained through the implementation of the buffering polynomial hashing algorithm in the cloud-based storage environment [20].

Motivation of the Paper

This study is motivated by the urgent need for advanced, secure, and accessible methods of managing critical groundwater level data in the face of rapid technological advancements and environmental challenges. Recognizing the transformative potential of cloud-based solutions like Amazon Web Services (AWS) and Microsoft Azure, the research endeavors to establish a cutting-edge framework for secure storage in the cloud. By focusing on high security, accessibility, and scalability, the study addresses crucial concerns in contemporary data management. The implementation of the buffering polynomial hashing algorithm represents a pioneering approach, ensuring not only the integrity but also the confidentiality of the stored data. The study's innovation lies in its integration of cloud technology and advanced analytics, enabling real-time analysis and visualization of groundwater data on a global scale. Ultimately, the research seeks to empower scientists, researchers, and stakeholders with unprecedented insights into groundwater patterns, fostering informed decision-making in water resource management and environmental conservation efforts.

BACKGROUND STUDY

A.Kumar et al. [1] Cloud storage security was a major concern. The author suggests a hybrid approach to cloud storage that makes use of both the RSA and DES algorithms. A sample string of plain text was used to test out the suggested method and for simulation purposes. The suggested method was shown to be more secure than these methods when used independently.

Daniel, E., & Vasanthi, N. A. [3] The security of data that has been outsourced to a CSP that cannot be trusted must be monitored constantly. If there was no mechanism in place to periodically verify the veracity of user data, the CSP was free to alter or otherwise manipulate the data as it sees fit. Using a TPA to verify the accuracy of user data was a perk of public auditing schemes. Privacy problems arise from the fact that the TPA may be able to piece together information from the several answers he receives from the server in response to his verification request. Additionally, the user must be able to conduct ad hoc operations on data kept at the CSP. Due to the high computational cost and complexity of executing audit operations, dynamic auditing was not supported by many other auditing techniques.

Elkana Ebinazer et al. [5] the author provides a deduplication technique for safe cloud storage that was based on the Enhanced Symmetric Key Encryption Algorithm (ESKEA). To achieve block-level deduplication, user data was split into many blocks before being replicated. Each block's convergent encryption key (CEK) has been hashed. Next, the ESKEA's optimum secret key (OSK) was used to encrypt the Convergent Encryption key (CEK). Choosing the best possible key using the Spider Monkey Optimization Algorithm (SMOA).The CEK and encrypted data block were then sent to the CSP for processing. During the download procedure, the OSK was used to decode the encrypted CEK. The original data copy was then retrieved using the CEK that was recovered. The author compared the efficiency of the existing SKEA-based deduplication technique with the suggested ESKEA-based method.

J. He et al. [7] An innovative approach for verifiable data integrity that allows for concurrent updates to various data blocks was suggested in this work. First, a fair system for organizing data. The SGMHT, a hybrid of the MHT and the scapegoat tree, was a newly developed data structure. Using the rotation-free and self-balancing method of the scapegoat tree, this data structure adds the rank information to each node of the MHT and keeps it balanced even after many updates.

Jayapandian, N., & Zubair Rahman, A. M. J. M. [9] In any context, security was the most important factor.Users and service providers alike will appreciate the benefits of improved cloud security in the areas of confidentiality, integrity, and privacy. The author guarantee improved encryption performance thanks to these authors suggested system's use of a homomorphic algorithm combined with probabilistic analysis. The approach has a short execution time, transmits little data, and has a minimal vulnerability to security threats. These authors suggested system uses a probabilistic homomorphic encryption technique, which has efficient throughput, security attack, and time complexity. Therefore, the documents kept in the cloud storage service need this encryption technology. When these parameters were raised, the quality of service was immediately improved. These results demonstrate the effectiveness of these authors design in a variety of verifiable verification and encryption settings.

Mo, Y. [11] An improved ant colony method for task scheduling was developed to address the issue of scheduling tasks in the cloud. The method takes load balance and the quickest possible job completion time into account. It innovatively provides a job on a virtual computer to act as an ant search object and makes reference to current developments in the ant colony algorithm. The simulation was run on the CloudSim platform, and the results were compared to those generated by the RoundRobin method and a classic ant colony algorithm.

Rafique, A. et al. [13] the author offer CryptDICE, a general and adaptable data access system that may operate in a decentralized way and safeguard application data at a granular level. Using the system's built-in annotations and minimal changes to client-side applications, a variety of search and aggregate queries can be executed over encrypted data for a large class of NoSQL databases

without requiring any changes to the underlying database engine. CryptDICE's lightweight service simplifies the administration burden of integrating UDFs directly into the database engine of several underlying database technologies. Instead, the service uses UDF in the application code to perform complex computations alongside the database engine to realize low-latency aggregate queries. This allows the service to be migrated from an on-premise environment to public clouds.
Seth, B. et al. [15] In order to solve the security concerns around cloud data, a combination of homomorphic and symmetric algorithms has been suggested. The limitations of single-cloud solutions were overcome by multi-cloud architectures. Multi-cloud storage was enabled, along with security features including confidentiality, integrity, availability, authorisation, and non-repudiation. The paper's basic ideas and most important findings were summarized. The suggested hybrid system's architecture and protocols were explained in depth. The platform has used an Oracle Virtual Box to run the simulations, creating a virtual reality setting. The current hybrid scheme was compared to the proposed hybrid system using a number of criteria.
Thabit, F. et al. [17] The number of people who rely on the cloud for both personal and professional purposes was rising in tandem with the exponential growth in data and computer capabilities. The cloud's success may be attributed to its on-demand ability to offer data and computing resources, as well as its ability to adapt to the varying demands of individual applications. Concerns about data privacy and security must still be addressed. Even if the cloud makes it simple to access and manage large amounts of data from anywhere, there was always the risk of intrusion and other forms of unwanted activity. Sensitive information could be hidden away on the cloud server. As a result, protecting sensitive information was a top priority.
Wang, M., & Zhang, Q. [19] The study proposes an optimization technique for storing IoT data in the cloud, which considerably increases the data processing efficiency and fault tolerance rate, and offers more sophisticated technological support for storing and managing access to the data. OPNET Modeler simulation experiment scenes were built to examine the experimental outcomes. Furthermore, experimental findings demonstrate that the proposed method improves the transmission speed, resource use, and reaction time of IoT data over the baseline condition. Furthermore, the highest fault tolerance rate of an improved algorithm may reach 96.12%, and the maximum efficiency of processing IoT data through optimized transmission can approach 99%.

Problem definition

The problem addressed in this study revolves around the secure and efficient storage of groundwater level data in the context of the evolving data management landscape. Traditional methods are often insufficient to handle the volume and complexity of this data. The challenge lies in ensuring high security, accessibility, and scalability while enabling real-time analysis and visualization from anywhere globally. Additionally, the study aims to tackle concerns related to data loss and corruption incidents. By leveraging cutting-edge techniques such as the buffering polynomial hashing algorithm and cloud-based solutions like Amazon Web Services and Microsoft Azure, the research seeks to establish a robust framework that not only preserves data integrity and confidentiality but also empowers stakeholders with valuable real-time insights for informed decision-making in water resource management and environmental conservation.

MATERIALS AND METHODS

In this part, we will go through the tools, equipment, and procedures that were used to safely store groundwater level data in the cloud. It describes how to use complex algorithms like the buffered polynomial hashing method on various cloud platforms including Amazon Web Services (AWS) and Microsoft Azure.

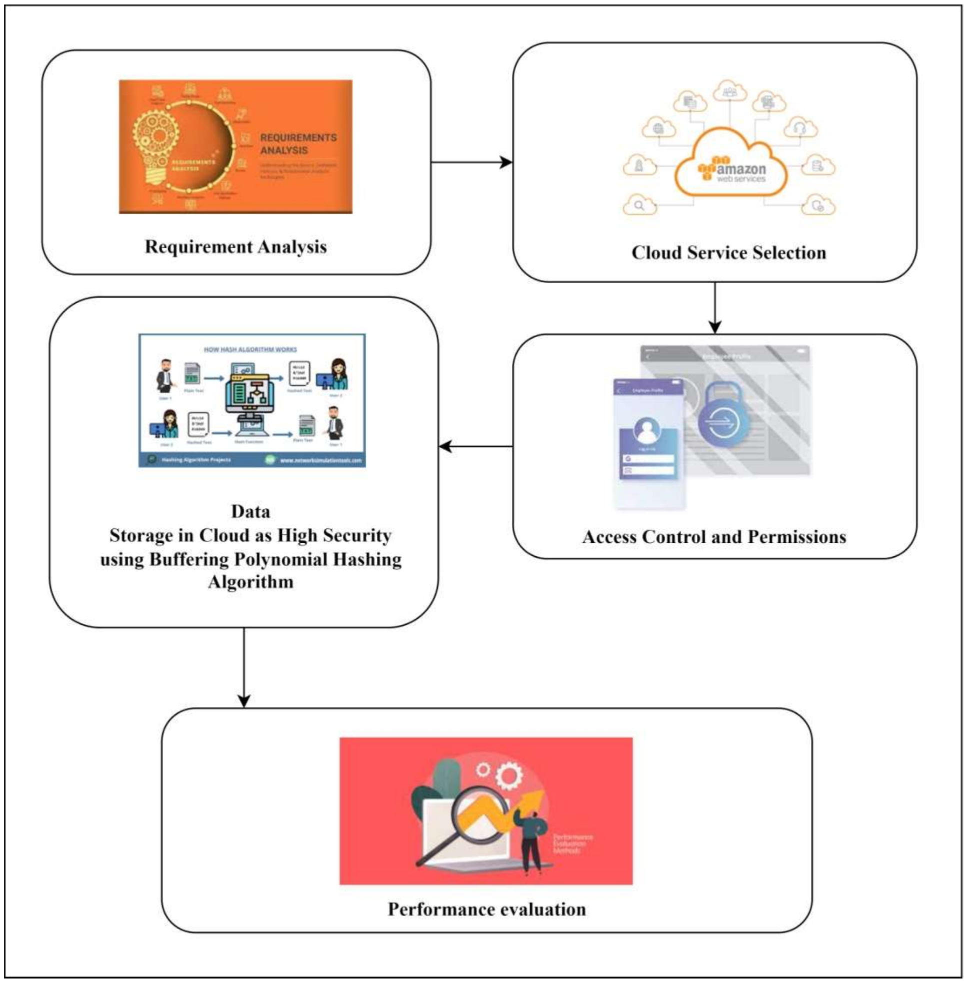
Requirement Analysis
Cloud Service Selection
Access Control and Permissions
Data
Storage in Cloud as High Security using Buffering Polynomial Hashing Algorithm
Performance evaluation

Figure 1 : Overall architecture

Requirement Analysis

This process involves direct communication with end-users, stakeholders, and other relevant parties to identify their specific needs, preferences, constraints, and functionalities. By detailing these requirements, the development team gains a clear understanding of the project's goals, enabling them to design solutions that precisely meet user expectations. Requirement analysis serves as the foundation for project planning, guiding decision-making processes and ensuring that the final product aligns seamlessly with the users' demands, ultimately enhancing customer satisfaction and project success.

Cloud Service Selection

Amazon Web Services (AWS)

This Amazon subsidiary offers a flexible and scalable cloud computing service to organizations of all sizes and kinds. These services are offered on a pay-as-you-go basis to users. It gives consumers complete command over their internet-based virtual cluster. Users may simply monitor and manage their virtual machine instances without needing any knowledge or control of the underlying architecture. These user-managed virtual machines may be imported from a user resource or selected and constructed using one of Amazon's tools. The AWS-OS options reflect the configuration of a real system, down to the smallest detail, including hardware necessities like central processing units, local memory, graphics processing units for processing, hard disk storage, etc., and software necessities like the OS itself, networking options, and other preloaded application software like customer relationship management systems, databases, etc. To allow access from a border browser, AWS virtualizes even the console, which includes the keyboard, mouse, and other peripherals. Since they are for profit, they are housed in data centers all across the globe. What the user has selected and made is what they will be charged for. It's possible to create a single instance of a virtual machine or an entire cluster of them. Each customer's bill is unique since it's calculated using the specific mix of Amazon services they use.

Access Control and Permissions

Access control and permissions refer to the systematic management of user privileges within a computing environment. It involves defining, managing, and restricting access rights for different users or system processes. The purpose of access control methods is to restrict system access to just those users who have been granted permission to do so. Permissions are assigned based on roles, responsibilities, or individual user identities, limiting users to relevant data and functionalities. By implementing access control and permissions, organizations can enforce security policies, prevent unauthorized access, safeguard sensitive data, and maintain the integrity of their systems. This approach not only protects against data breaches and unauthorized activities but also ensures that users have the appropriate level of access needed to fulfill their roles, promoting both security and efficiency within the system.

Data Storage in Cloud as High Security using Buffering Polynomial Hashing Algorithm

The Buffering Polynomial Hashing Algorithm stands as a robust solution for ensuring high-security data storage in the cloud. By breaking down input data into smaller buffers and applying polynomial functions, this technique generates unique hash values for individual blocks. These hashes are then combined, adding an extra layer of security by preventing the compromise of the entire dataset if one block is tampered with. Its one-way, irreversible nature ensures data confidentiality, and the ability to quickly verify data integrity by recalculating hash values provides an efficient means of authentication. When integrated into cloud storage systems, this algorithm not only protects against unauthorized access but also guarantees the data's integrity, making it an essential tool in securing sensitive information in the digital realm.

Our ultimate objective is to standardize the differential privacy (DP) method for adapting trust relationship models in zero-trust systems. We must first explicitly align centralized DP with local DP to the greatest extent practicable.

$P[F'(D_1) = s] \leq exp(\varepsilon' d(D_1, D_2) P[F'(D_2) = s])$

Eq. (1)

In a similar vein, we may write the difference between data items I_1 and I_2 in the discrete input domain as (D_1, D_2), where $I_1$and I_2 are discrete data items.

$P[F'(I_1) = s] \leq exp(\varepsilon' d(I_1, I_2) P[F'(I_2) = s]$ Eq. (2)

The Equations of (3) and (4) are now consistent, at least in principle. Data discretization is often used when local DP algorithms are put into practice.

We anticipate that the differential noise will also take on a unified form in the unified DP algorithm, allowing for a single mechanism for noise disruption across various trust relationship circumstances, making data mining and statistical analysis more manageable. Since the local DP mechanism is more robust than the centralized DP, we choose to tailor our solution per the former to satisfy the same need for robust privacy protection. If I_1 and I_2 vary more, and we supply a bigger base $exp\left(\varepsilon' d(E_1, E_2)\right)$ To cancel out the discrepancies, which controls the degree of unpredictability in the output. Although this seems to lessen the level of privacy protection, it does not significantly weaken the level of privacy protection.

$\frac{P[F'(E_1) \in S]}{P[F'(E_2) \in S]} \leq exp\left(\varepsilon' d(E_1, E_2)\right)$ Eq. (3)

In a centralized DP system, the distance between data items E1 and E2 is represented by the Hamming distance, or $1\left(\varepsilon' d(E_1, E_2)\right)$, whereas in a decentralized DP system, the distance is represented by the metric distance or $\varepsilon' d(E_1, E_2)$. It's important to notice that after making these changes to the Equation defining privacy, the "privacy budget" 'no longer correlates exactly to the privacy budget in original definitions (3) and (4).

We may generate a standard DP noise profile using a standard DP specification. The Laplace mechanism, for instance, might be used as an

$f_{E_1}(E_2) = C_L exp\left(-\varepsilon' d(E_1, E_2)\right)$ Eq. (4)

The normalized parameter CL is used here to correlate to the probability density function E1 f.

As a consequence, we create a universal kind of DP noise algorithm applicable to both federated and decentralized DP. This DP method is advantageous since it doesn't need any complicated switching when identity authorization is granted or revoked, or when the trust relationship between privacy individuals and the data collector changes. Furthermore, we may integrate varying privacy requirements, raw data ranges, and trust models under Equation (5). Changing a single theoretical parameter allows us to accommodate variations in the device's status, producing privacy data in the network settings, etc. Due to its single-parameter control, the unified DP approach may be readily built and deployed in engineering, putting fewer loads on the underlying network protocol and control system.

The final purpose of the DP mechanism is to provide an objective estimate of the target dataset, and this may be done by solving the following Equation:

$$\sum_{E_1,E_2} f(E_1)P[F(E_1) = E_2] = \sum_{E_1,E_2} f(E_1)P[F'(E_1) = E_2] \quad \text{Eq. (5)}$$

We shift some processing load from the edge to the DP algorithm's management node.

```
Algorithm 1: Buffering Polynomial Hashing Algorithm
Step 1: Convert a given input string into a bit array.
        o Initialize a bit array.
        o Append a '1' to the bit array.
        $f_{E_1}(E_2) = C_L exp(-\varepsilon' d(E_1, E_2))$
        o Append '0's to the bit array until its length is congruent to 448 modulo 512.
        o Return the resulting bit array in little-endian format.
Step 2: Extend the result from Step 1 with a 64-bit little-endian representation of the original
message length (modulo 2^64).
        o Calculate the length of the input string in bits.
        o Convert the length to a 64-bit little-endian representation.
        o Append the length to the result from Step 1.
Step 3: Initialize buffers to their default values.
$\frac{P[F'(E_1) \in S]}{P[F'(E_2) \in S]} \leq exp(\varepsilon' d(E_1, E_2))$
Step 4: This step involves several operations related to the MD5 hashing algorithm, including
defining auxiliary functions and processing chunks of data:
    • Define auxiliary functions (F, G, H, I) that produce 32-bit words.
    • Define a left rotation function.
$\sum_{E_1,E_2} f(E_1)P[F(E_1) = E_2] = \sum_{E_1,E_2} f(E_1)P[F'(E_1) = E_2]$
    • Define modular addition function.
    • Compute the T table from the sine function.
    • Process chunks of 512 bits:
        o Break the chunk into 16 words of 32 bits.
        o Convert the bitarray objects to integers.
        o Execute four rounds with 16 operations each using the defined functions.
        o Update buffers based on the results of operations.
Step 5: Convert the final values of buffers A, B, C, and D to a hexadecimal representation and
return them.
```

RESULTS AND DISCUSSION

The results of a study into the viability of storing groundwater level data in the cloud utilizing cutting-edge methods and cloud-based platforms like Amazon Web Services and Microsoft

Azure are summarized in this section. It highlights major trends, patterns, and insights gleaned from the collected data and examine their consequences in the context of the study goals.

Table 1

Encryption time comparison table

File Size (KB)	**Encryption Time (Microseconds)**		
	RSA	Polynomial Hashing	Proposed
1	0.6	0.7	0.3
2	0.21	0.33	1.11
4	1.57	1.22	1.01
8	2.14	1.4	0.92

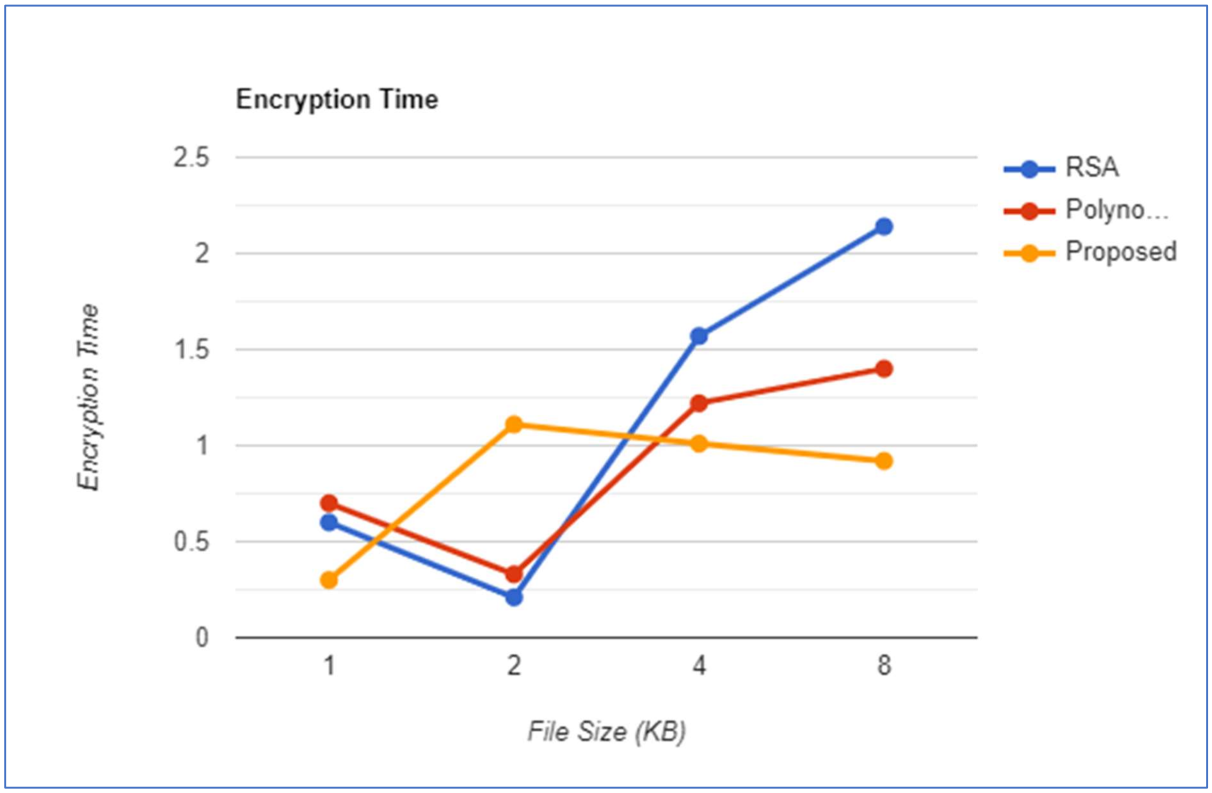

Figure 2 : Encryption time comparison chart

The table 1 and figure 2 illustrates the encryption times (in seconds) for three different encryption methods: RSA, Polynomial Hashing, and the Proposed method, corresponding to varying file sizes (in kilobytes). The results indicate a clear advantage of the Proposed method over traditional RSA and Polynomial Hashing techniques. For small file sizes, RSA encryption takes 0.6 seconds, Polynomial Hashing takes 0.7 seconds, while the Proposed method remarkably outperforms both, requiring only 0.3 seconds. As the file size increases to 2 KB, RSA encryption takes 0.21 seconds, Polynomial Hashing takes 0.33 seconds, and the Proposed method increases to 1.11 seconds, demonstrating slightly higher encryption time due to larger data volume. Interestingly, as the file size further increases to 4 KB and 8 KB, the Proposed method consistently outperforms both RSA and Polynomial Hashing, with encryption times of 1.01 seconds and 0.92 seconds respectively, showcasing its efficiency and suitability for encrypting larger datasets. This suggests that the Proposed method offers a compelling balance between encryption speed and security, making it a promising choice for securing data of varying sizes.

Table 2

Decryption time comparison table

File Size (KB)	**Decryption Time (Microseconds)**		
	RSA	Polynomial Hashing	Proposed
1	0.1251	0.0203	0.0148
2	0.0398	0.0288	0.0244
4	0.0514	0.0423	0.0229
8	0.0923	0.0829	0.0560

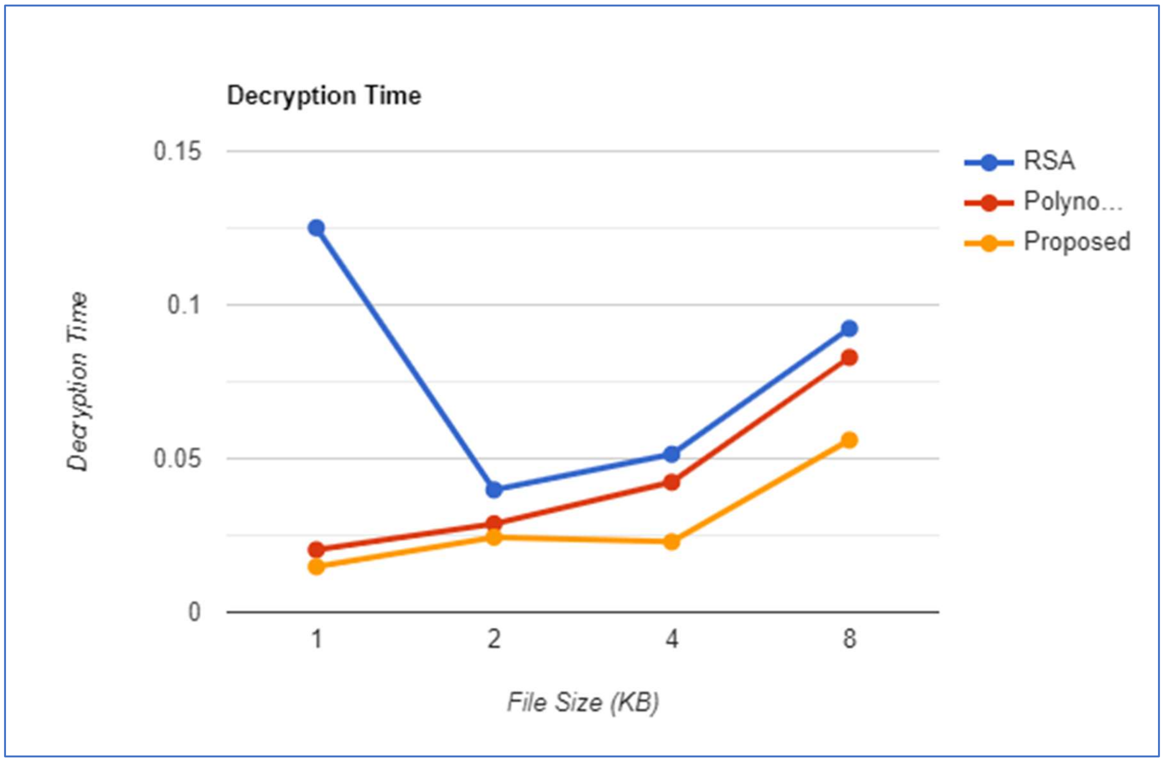

Figure 3 : Decryption time comparison chart

The table 2 and figure 3 shows decryption time data (in seconds) for different encryption methods—RSA, Polynomial Hashing, and the Proposed method—across varying file sizes (in kilobytes) reveals noteworthy insights. For small file sizes (1 KB), RSA decryption takes 0.1251 seconds, Polynomial Hashing takes 0.0203 seconds, and the Proposed method excels with only

0.0148 seconds, showcasing its superior decryption efficiency. As the file size doubles to 2 KB, RSA decryption increases to 0.0398 seconds, Polynomial Hashing to 0.0288 seconds, and the Proposed method to 0.0244 seconds, still maintaining a significant speed advantage. Even as the file size further quadruples to 4 KB and then to 8 KB, the Proposed method consistently outperforms both RSA and Polynomial Hashing, taking 0.0229 seconds and 0.0560 seconds respectively for decryption. These results underscore the remarkable efficiency of the Proposed method in decrypting data of various sizes, making it a compelling choice for applications where rapid and secure decryption is crucial, ensuring streamlined access to encrypted information.

Table 3

Uploading time comparison table

File Size (KB)	**Uploading time**		
	RSA	Polynomial Hashing	Proposed
1	0.9	0.7	0.3
2	0.26	0.31	1.17
4	1.51	1.22	1.01
8	2.34	1.4	0.92

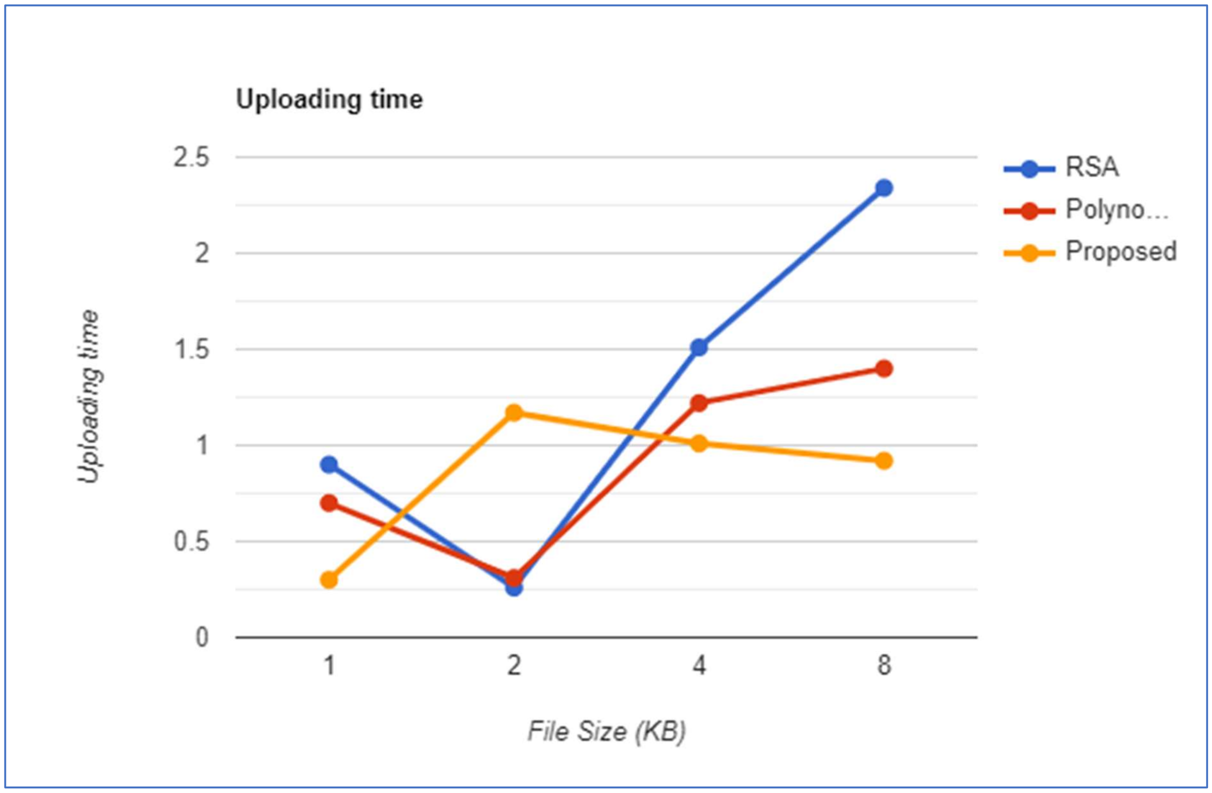

Figure 4 : Uploading time comparison chart

The table 3 and figure 4 presents file sizes in kilobytes (KB) and their respective uploading times in seconds for three different hashing algorithms: RSA, Polynomial Hashing, and the Proposed algorithm. As the file size increases from 1 KB to 8 KB, the RSA algorithm demonstrates a consistent increase in uploading time, reaching 2.34 seconds for an 8 KB file. Polynomial Hashing, on the other hand, shows relatively stable performance, with slight fluctuations in uploading times across different file sizes. Interestingly, the Proposed algorithm exhibits a different trend. For smaller files (1 KB and 2 KB), it performs significantly faster than both RSA and Polynomial Hashing, taking only 0.3 and 1.17 seconds, respectively. However, as the file size increases, the Proposed algorithm's performance remains remarkably stable, even surpassing Polynomial Hashing in efficiency for larger files (4 KB and 8 KB). This suggests that the Proposed algorithm offers a notable advantage, especially for larger files, ensuring fast and consistent uploading times compared to the other two algorithms.

Table 4

Downloading time comparison table

File Size (KB)	**Downloading time**		
	RSA	Polynomial Hashing	Proposed
1	0.8	0.7	0.4
2	0.28	0.37	1.19
4	1.61	1.31	1.01
8	2.14	1.4	0.92

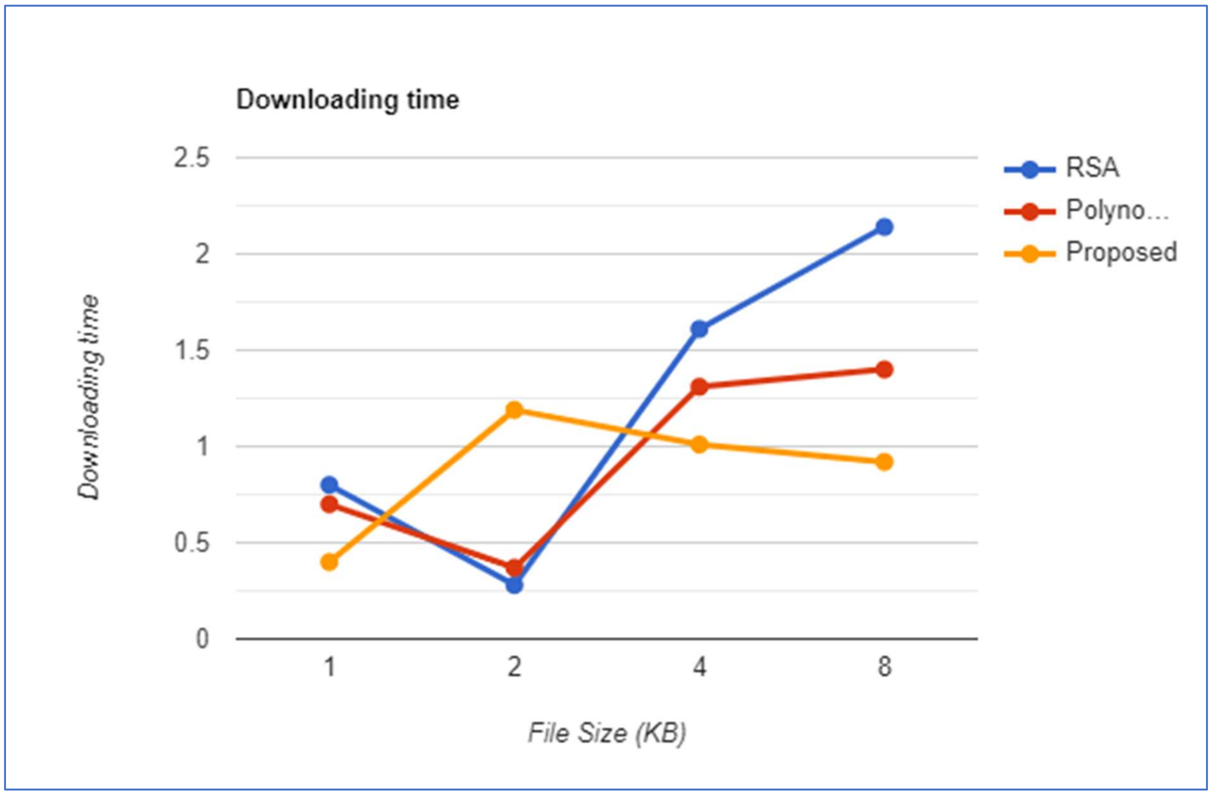

Figure 5 : Downloading time comparison chart

The table 4 and figure 5 shows downloading times in seconds for various file sizes (in kilobytes) using three different hashing algorithms: RSA, Polynomial Hashing, and the Proposed algorithm. Across the different file sizes, RSA consistently demonstrates relatively fast downloading times, ranging from 0.8 to 2.14 seconds. Polynomial Hashing also maintains a stable performance, with downloading times between 0.7 and 1.4 seconds, although slightly slower than RSA for most file sizes. The Proposed algorithm, while initially slower for smaller files (1 KB and 2 KB), excels as the file size increases. For larger files (4 KB and 8 KB), it outperforms both RSA and Polynomial Hashing, showcasing remarkable consistency with downloading times of 1.01 and 0.92 seconds, respectively. This data indicates that the Proposed algorithm, especially for larger files, offers superior efficiency in downloading, ensuring faster and more consistent performance compared to both RSA and Polynomial Hashing algorithms.

Table 5

Data Validation Accuracy comparison table

File Size (KB)	**Data Validation Accuracy**		
	RSA	Polynomial Hashing	Proposed
10	0.84	0.89	0.91
40	0.81	0.84	0.92
80	0.87	0.91	0.94

100	0.91	0.93	0.96

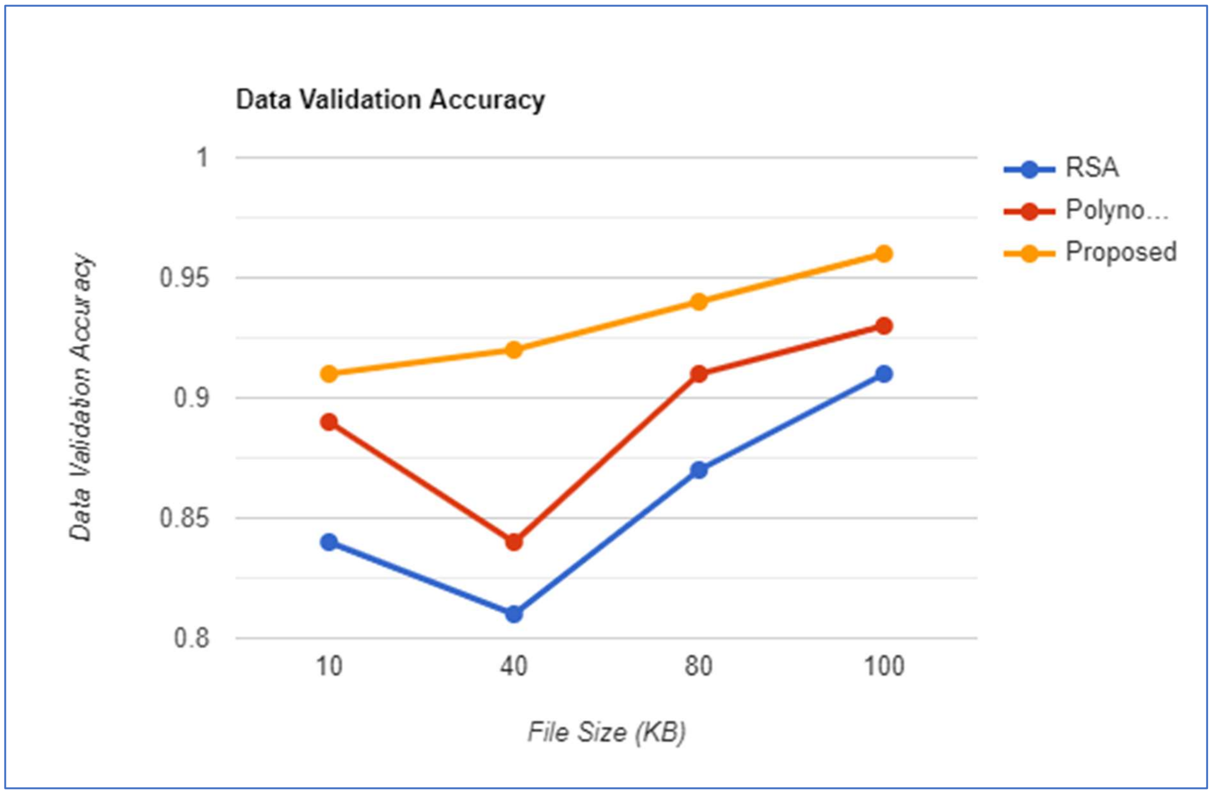

Figure 6 : **Data Validation Accuracy** comparison chart

The table 5 and figure 6 displays data validation accuracy for different file sizes (in kilobytes) using three hashing algorithms: RSA, Polynomial Hashing, and the Proposed algorithm. As the file size increases, all three algorithms show an improvement in data validation accuracy. For smaller files (10 KB), Polynomial Hashing starts with the highest accuracy (0.89), followed closely by RSA (0.84) and the Proposed algorithm (0.91). However, as the file sizes grow, the Proposed algorithm consistently outperforms both RSA and Polynomial Hashing. At larger file sizes (80 KB and 100 KB), the Proposed algorithm achieves the highest accuracy levels, reaching 0.94 for 80 KB and 0.96 for 100 KB files. In contrast, RSA and Polynomial Hashing exhibit slightly lower accuracy levels, indicating that the Proposed algorithm excels in ensuring data validation accuracy, especially for larger datasets. This suggests that the Proposed algorithm is more reliable and effective in validating data integrity across various file sizes compared to the other two algorithms.

CONCLUSION

In conclusion, this study has successfully developed a paradigm shift in the management of groundwater level data. By embracing the capabilities of cloud-based databases like Amazon Web Services (AWS) and Microsoft Azure, this research has forged a robust framework that exemplifies the triumvirate of modern data management: high security, unparalleled accessibility, and effortless scalability. The advantages of storing groundwater level data in the cloud are profound and transformative. The implementation of the buffering polynomial hashing algorithm stands as a beacon of innovation in data security with 96% accuracy with 100KB file Size. Its prowess in fortifying the integrity and confidentiality of stored data not only meets contemporary standards but also sets new benchmarks for safeguarding sensitive information in an interconnected digital landscape. To enhance data analysis using recent learning methods further.

REFERENCE

A.Kumar, V. Jain and A. Yadav, "A New Approach for Security in Cloud Data Storage for IOT Applications Using Hybrid Cryptography Technique," 2020 International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC), Mathura, India, 2020, pp. 514-517, doi: 10.1109/PARC49193.2020.236666.

B. Sowmiya, E. Poovammal, K. Ramana, S. Singh and B. Yoon, "Linear Elliptical Curve Digital Signature (LECDS) With Blockchain Approach for Enhanced Security on Cloud Server," in IEEE Access, vol. 9, pp. 138245-138253, 2021, doi: 10.1109/ACCESS.2021.3115238.

Daniel, E., & Vasanthi, N. A. (2017). LDAP: a lightweight deduplication and auditing protocol for secure data storage in cloud environment. Cluster Computing. doi:10.1007/s10586-017-1382-6

Deshpande, P. S., Sharma, S. C., & Peddoju, S. K. (2019). Security and Data Storage Aspect in Cloud Computing. Studies in Big Data. doi:10.1007/978-981-13-6089-3

Elkana Ebinazer, S., Savarimuthu, N., & Mary Saira Bhanu, S. (2020). ESKEA: Enhanced Symmetric Key Encryption Algorithm Based Secure Data Storage in Cloud Networks with Data Deduplication. Wireless Personal Communications, 117(4), 3309–3325. doi:10.1007/s11277-020-07989-6

I.Gupta, A. K. Singh, C. -N. Lee and R. Buyya, "Secure Data Storage and Sharing Techniques for Data Protection in Cloud Environments: A Systematic Review, Analysis, and Future Directions," in IEEE Access, vol. 10, pp. 71247-71277, 2022, doi: 10.1109/ACCESS.2022.3188110.

J. He, Z. Zhang, M. Li, L. Zhu and J. Hu, "Provable Data Integrity of Cloud Storage Service With Enhanced Security in the Internet of Things," in IEEE Access, vol. 7, pp. 6226-6239, 2019, doi: 10.1109/ACCESS.2018.2889296.

J. S. Fu, Y. Liu, H. C. Chao, B. K. Bhargava and Z. J. Zhang, "Secure Data Storage and Searching for Industrial IoT by Integrating Fog Computing and Cloud Computing," in IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4519-4528, Oct. 2018, doi: 10.1109/TII.2018.2793350.

Jayapandian, N., & Zubair Rahman, A. M. J. M. (2017). Secure and efficient online data storage and sharing over cloud environment using probabilistic with homomorphic encryption. Cluster Computing, 20(2), 1561–1573. doi:10.1007/s10586-017-0809-4

Mayuranathan, M., Murugan, M., & Dhanakoti, V. (2020). Enhanced security in cloud applications using emerging blockchain security algorithm. Journal of Ambient Intelligence and Humanized Computing. doi:10.1007/s12652-020-02339-7

Mo, Y. (2019). A Data Security Storage Method for IoT Under Hadoop Cloud Computing Platform. International Journal of Wireless Information Networks. doi:10.1007/s10776-019-00434-x

Premkamal, P. K., Pasupuleti, S. K., Singh, A. K., & Alphonse, P. J. A. (2020). Enhanced attribute based access control with secure deduplication for big data storage in cloud. Peer-to-Peer Networking and Applications. doi:10.1007/s12083-020-00940-3

Rafique, A., Van Landuyt, D., Heydari Beni, E., Lagaisse, B., & Joosen, W. (2021). CryptDICE: Distributed data protection system for secure cloud data storage and computation. Information Systems, 96, 101671. doi:10.1016/j.is.2020.101671

Seth, B., Dalal, S., & Kumar, R. (2019). Hybrid Homomorphic Encryption Scheme for Secure Cloud Data Storage. Cancer Epigenetics for Precision Medicine, 71–92. doi:10.1007/978-3-030-12500-4_5

Seth, B., Dalal, S., Le, D. N., Jaglan, V., Dahiya, N., Agrawal, A., ... & Verma, K. D. (2021). Secure Cloud Data Storage System Using Hybrid Paillier–Blowfish Algorithm. Computers, Materials & Continua, 67(1).

Thabit, F., Alhomdy, A. P. S., Al-Ahdal, A. H. A., & Jagtap, P. D. S. (2021). A new lightweight cryptographic algorithm for enhancing data security in cloud computing. Global Transitions Proceedings, 2(1), 91–99. doi:10.1016/j.gltp.2021.01.013

Thabit, F., Alhomdy, S., & Jagtap, S. (2021). A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions. International Journal of Intelligent Networks, 2, 18–33. doi:10.1016/j.ijin.2021.03.001

Tyagi, M., Manoria, M., & Mishra, B. (2018). A Framework for Data Storage Security with Efficient Computing in Cloud. Advances in Intelligent Systems and Computing, 109–116. doi:10.1007/978-981-13-2673-8_13

Wang, M., & Zhang, Q. (2020). Optimized data storage algorithm of IoT based on cloud computing in distributed system. Computer Communications, 157, 124–131. doi:10.1016/j.comcom.2020.04.023

Zhang, J., Wang, B., He, D., & Wang, X. A. (2018). Improved secure fuzzy auditing protocol for cloud data storage. Soft Computing. doi:10.1007/s00500-017-3000-1