International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 www.ijisae.org Original Research Paper

An Integrated M-Tree Approach for Securing Information in Public Auditing Clouds

[1*]Banoth Anantharam , [2]Dr.Rajesh Sharma, [3]Dr. B.Kavitha Rani

Submitted: 20/10/2023 **Revised**: 18/12/2023 **Accepted**: 26/12/2023

Abstract: In cloud computing one of the most important components is "IAM", Identity and Access management tool to decide the rights of an individual over a resource. The identity is a property, which contains most sensitive and vital information in the encrypted format. Access represents the parameters like viewing and types of actions that a user can perform on the data. In general, integrity of the data is provided by the Third-Party Auditor (TPA) by generating the hash values for the encrypted blocks. It takes lot of time to audit every encrypted block by the TPA, this increases the cost of deployment. The cloud resources to provide the access management to the resources either use the regenerating code whenever it is being shared with others or using the proxy server. The proxy servers suffer from "SSL stripping", i.e., an attacker hacks the victim's information by sending the https URL. The proposed research aims to solve this problem by enhancing the Merkle Trees technique in which the blocks are secured using the hashing functions. These hashing functions are generated by combining the properties of MD5 and SHA-256. The proposed research combined these two algorithms because while transferring the information the communications should be faster which is served by MD5 whereas security should not be breached, i.e., data should be encrypted in short and more secured format, this is achieved by SHA-256.

Keywords: *Message Digest, Hashing, Encryption, SSL Stripping, Proxy Server, Third Party Auditor*

1. INTRODUCTION

Privacy is the most crucial element while sharing the data between two different parties.

1.1 Privacy Preservation Techniques: Because it guarantees that only those with authorization can access data transmitted in the cloud, privacy is crucial in cloud computing. It would not be possible to guarantee the integrity of cloud-stored data without appropriate data privacy. In order to lessen security risks and other features of the data owner, privacy preservation attempts to secure the data. There are many challenges that give rise to privacy concerns, particularly when the data holder stores their data in an environment without centralised management [1]. The distribution of data owners' degrees and the detection of sensitive features in the data list are two issues with privacy mechanisms. Security for data processing and storage are two components that help protect privacy in cloud systems. Types of privacy preservation techniques

Privacy preserving search in data clouds: Before utilising homomorphic encryption, the data is hidden. This method encrypts and packages cloud data using usage policy. The data access mechanism evaluates the data environment's trustworthiness, consults its policy, and builds a virtualization environment [2].

Anonymity-based method: Before being stored in the cloud, sensitive data is anonymized. This anonymity method will process the microdata before they are made public, after which deliver this anonymized data to cloud service providers. After that, the network operator can incorporate the auxiliary data and analyse the anonymous data to mine the desired knowledge.

Privacy-preserved authentication: Sharing of private data with creation of anonymous IDs. Prior to communication, source authentication verified the sender nodes' credibility. Bilinear pairings are provided for identification, together with ciphertext-policy attribute-based access control. To keep communications dynamic, dual session identities and pseudorandom numbers are added as session variation operators.

[1*]Research Scholar, Department of Computer Science and Engineering, School of Engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore(M.P),India.
Email ID: ananth502.ram@gmail.com
[2] Department of Computer Science and Engineering, School of Engineering, Sri Satya Sai University of Technology and Medical Sciences, Sehore(M.P),India.
Email ID: drsharma.sssutms@gmail.com
[3]Department of Computer Science and Engineering, CMR Technical Campus, Hyderabad, Telangana, India.
Email ID: phdknr1@gmail.com

Privacy preserving data computation: Computing with many parties' data without revealing each party's individual data Cloud providers can be parties.

Privacy Preserving Data Outsourcing: ensures privacy by expanding the data to unrestricted areas that are practical for the inputs. The gateway is compelled to duplicate unencrypted data storage, re-implement some cloud services, and reverse-engineer them, which defeats the point of outsourcing data and compute.

Privacy-preserved Access Control: establishes user access privileges and implements access control. The roles are granted access to the tuples under the approved predicate thanks to an access control policy.

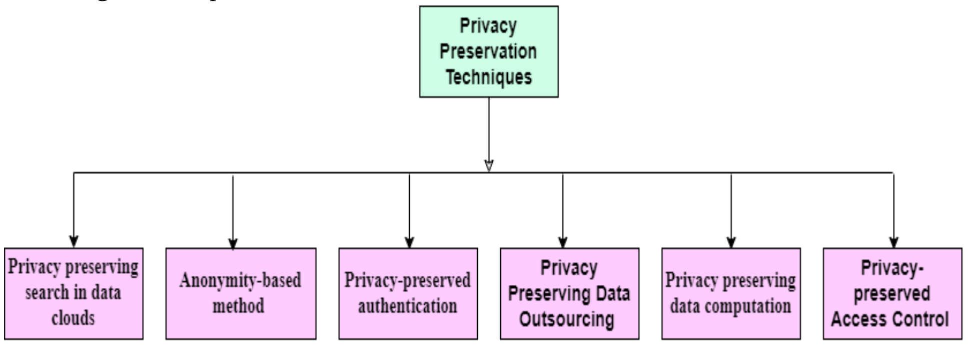
Privacy Preservation Techniques
Privacy preserving search in data clouds
Anonymity-based method
Privacy-preserved authentication
Privacy Preserving Data Outsourcing
Privacy preserving data computation
Privacy-preserved Access Control

Figure 1: Classification of Privacy Preservation Techniques

Research Motivation:

- **Efficient Data Sharing**: With cloud computing becoming increasingly prevalent, data sharing stands out as a pivotal feature. The ability to safeguard data while enabling dynamic access to cloud resources can enhance the adoption and utility of public cloud services.
- **Security Challenges**: Current security mechanisms in the cloud, such as cipher and identity-based approaches, have shown vulnerabilities, especially in authentication processes during resource revocation in groups. This presents a need for more robust security solutions.
- **Optimized Encryption**: The conventional method of providing large encryption blocks is resource-intensive. There is a potential for more efficient memory utilization through the implementation of compact hashing techniques.

1.2. Problems with regenerating the code for auditing scheme:

Because only the data owner is permitted to check the data's integrity and repair a malfunctioning server under the current architecture, only public auditing is intended. When taking into account the enormous quantity of users' forced resource capabilities and outsourced data, the task of repair and auditing in the cloud is really very expensive or pricey for users. The current auditing system demonstrates how difficult it is for consumers to always be online. TPA has access to the owner's data and can read and modify its contents. Although data integrity is not achieved as data is maintained by TPA, liability is increased. The third party auditor's access to the system and data is restricted due to legal limitations. Therefore, it is necessary to create a system that provides trustworthy cloud services for cloud customers as well as a solution to the regeneration problem.

1.3. Third Party Auditor: CSUs are thought of as taking a cost-effective strategy when they delegate the management of the encryption keys to a reliable TPA. In addition hand, the TPA is in charge of looking over the integrity of the data that has been outsourced on behalf of CSUs and managing the encryption keys. This will eliminate many of the issues with cloud data encryption by enabling efficient key management and guaranteeing data integrity. Additionally, the CSUs are not required to have a lot of computational capacity on their devices, which could be required to handle encryption keys, conduct data encryption, and authenticate messages for data that has been outsourced. Therefore, the usage of TPA enables every CSU to access the external data and take use of the true advantages of cloud computing using simple mobile devices. Although it will verify the accuracy of the data, the danger to the data owner's data integrity is increased. Since TPA has the ability to read and edit data, a means should be made available to address this issue.

1.3.1. Problems with Third Party Auditor: In the context of cloud computing, data auditing is used to address secure data storage. A TPA or the user themselves (the data owner) can perform auditing, which is the process of verifying user data. It aids in preserving the integrity of cloud-stored data. An organisation that is utilised to act on the client's behalf is the TPA. It decreases the client's overhead while also possessing all the expertise, knowledge, and professional skills required to execute the integrity verification process. Without obtaining a local copy of the data, TPA must effectively audit the cloud storage of data. It shouldn't know anything about the information kept on the cloud server. It shouldn't increase the cloud user's online workload in any way. failed to protect data confidentially. Because TPA is an unreliable party with the potential to be a hostile user or to spread insider threats, it must be taken seriously.

1.3.2. Need of Proxy Server: A proxy server is a device that allows clients' requests for particular apps or requests for particular resources to communicate with a particular server. There are many different types of proxies available and employed, depending on the purpose of a request made by a user to a server. Proxy servers' main objective is to safeguard the direct connection between internet resources and online users. The network device also prevents the disclosure of the client's IP address when a client submits a query to any other servers. It is simple to track and restrict the packet headers and payloads of requests made to social media websites by user nodes on the internal server. Proxy facilitates quick access to frequently visited material while storing the content of websites.

1.4 Research Contribution:

1. The proposed model constructed a M-Tree block for providing privacy to the resources in the cloud.

2. The M-Tree is extended by hashing the blocks with the integration of MD5 and SHA256. MD5 helps the model to store the larger chunks of data in smaller variables. SHA256 never enters into collision phase like other hashing algorithms.

3. The M-Tree helps the cloud to divide the blocks into halves and makes the process to complete at the faster rate.

The paper is organized in 5 sections to present the M-Tree security provision in the public clouds. The introduction section discusses about the types of privacy preserving techniques along with their drawbacks and it also discusses about the need of proxy server to provide security. The literature section discusses about the merits and demerits in the exiting approaches in providing security to the clouds. From the demerits it has analyzed the gaps and identified the advanced techniques to enhance the security. The proposed model extended the M-Tree by integrating the MD5 and SHA-256 to generate the encryption for every block. The advantage of this technique lies in less compaction format to store the data and need high end decryption techniques to hack the cloud. The results are discussed based on the encryption and decryption time along with memory utilization and response time from cloud services. Finally, the conclusion section presented the drawbacks of the TPA and need of integrated encryption to preserve the privacy in cloud.

2. LITERATURE SURVEY

Jian Liu et al [3]: Cloud data security is paramount. A solution based on RGC-based storage regenerates authenticators in the absence of data owners, involving multiple attributes like owner, proxy, and cloud. The process involves steps like setup, audit, and repair, ensuring data security during communication and storage.

Rinku Kumar et al [4]: Handling massive data, especially with security concerns, is challenging. A cryptographic encryption method was introduced which leverages AES/DES for data block encryption. This system efficiently manages data transmission among user, TPA, proxy, and cloud.

Mei Ping Liu et al [5]: Highlighted are two key challenges: forgery and redundant data storage. Through a specific setup and auditing process, data integrity is maintained. The system uses binary flags to detect data presence or absence and employs counter-measures against malicious data modifications.

Ainampudi Lavanya Satya et al [6]: Privacy issues and data protection are addressed using BLS signatures and homomorphic authenticators. The system generates security keys, ensuring data privacy and efficiency during transmission and storage, especially between the cloud, data owner, and authorized user.

S. Vaishnavee et al [7]: The focus is on enhancing cloud security. A new proxy approach is proposed for traditional audits. Data is encrypted, verified across multiple cloud servers, and repaired using specific strategies, all while ensuring high efficiency.

S. Nalini Poornima et al [8]: The research aims to improve user access security in cloud storage. The proposed system strengthens data protection during transmission, especially between data owner, TPA, and the cloud, ensuring optimal performance and security.

Yong Yu et al [9]: This research emphasizes user identity and data access security. It introduces a system where data properties help in construction results. The system ensures efficient data storage and access, emphasizing user identity and private key security.

Tessy Vincent et al [10]: Data regeneration, especially for corrupted files, is addressed. The proposed system uses proxies to aid in regeneration, combined with specific encryption techniques, ensuring data integrity and user convenience.

Surmila Thakchom et al [11]: The research identifies challenges in user data access. It presents a system that checks data dynamically, ensuring integrity and security. Several operations and protocols are introduced, with a focus on communication and computation cost efficiency.

Xin Tang et al [12]: The paper focuses on data integrity during audit processes. It introduces a key exchange scheme and a simplified mechanism for the protocol. A two-phase reversible watermarking method is used, ensuring data integrity, privacy, and efficiency.

Table 1: Overview of Existing Techniques

Author	Algorithm	Advantages	Limitations
Jian Liu et al [3]	RGC	Efficient regeneration code generation.	TPA scalability concerns with large datasets.
Rinku Kumar et al [4]	Cryptographic encryption	Easy data identification through gen values.	Data security concerns despite proxy audit verification.
Mei-Ping Liu et al [5]	Countermeasure technique	Key generation facilitated by countermeasure.	Data compromises by attackers aren't automatically detected.
Ainampudi Lavanya Satya et al [6]	BLS signature	High data encryption.	Data remains unprotected under certain risks.
S. Vaishnavee et al [7]	Public auditing	AES method offers efficient authorized schemes.	Performance decreases with large datasets.
S. Nalini Poornima et al [8]	SCSS	Security codes enhance performance during data upload.	Data security concerns during encryption checks.
Yong Yu et al [9]	Identity-related remote data	Enhanced data privacy using two properties.	Access issues if the private key is lost.
Tessy Vincent et al [10]	RGC	Improved data retrieval using Hash function.	Performance improvement opportunities exist.
Surmila Thakchom et al [11]	Bilinear Maps & BBS scheme	Reduced communication and computation costs.	Opportunities for enhancing data integrity operations.
Xin Tang et al[12]	RTIA	High-performance watermarking.	Verification is weak for data with high integrity.

3. PROPOSED METHODOLOGY

The proposed methodology uses the Integrated M-Trees technique to provide security the blocks while sharing the information. Hashing mechanisms provide more security than the encryption because the hash keys are more protected and more durable than the cipher text. The proposed research aims to integrate MD5 and SHA-256 algorithms and generate M-Trees

3.1. Hashing

Hashing is a process of taking input data of arbitrary size and producing a fixed-size output called a hash, which represents a unique and deterministic digital fingerprint of the entered data. Hashing has many applications in computer science and information security. One of the most common applications is data integrity checking. For example, when downloading a large file, a user can download the hash value of the file from a trusted source and then compute the hash value of the downloaded file to ensure that it matches the original hash value. If the hash values match, it means that the downloaded file is the same as the original file and that the data has not been corrupted or modified during transmission. Another important application of hashing is in password storage. Instead of storing passwords in plaintext, which can be easily compromised if a system is breached, passwords can be hashed and the resulting hash values can be stored instead. When a user logs in, the system can hash the entered password and compare it with the stored hash value to authenticate the user. There are many different hash functions available, including cryptographic hash functions such as SHA-256, MD5[13], and others, as well as non-cryptographic hash functions such as CRC32 and Adler-32. The process of providing privacy is presented in figure 2.

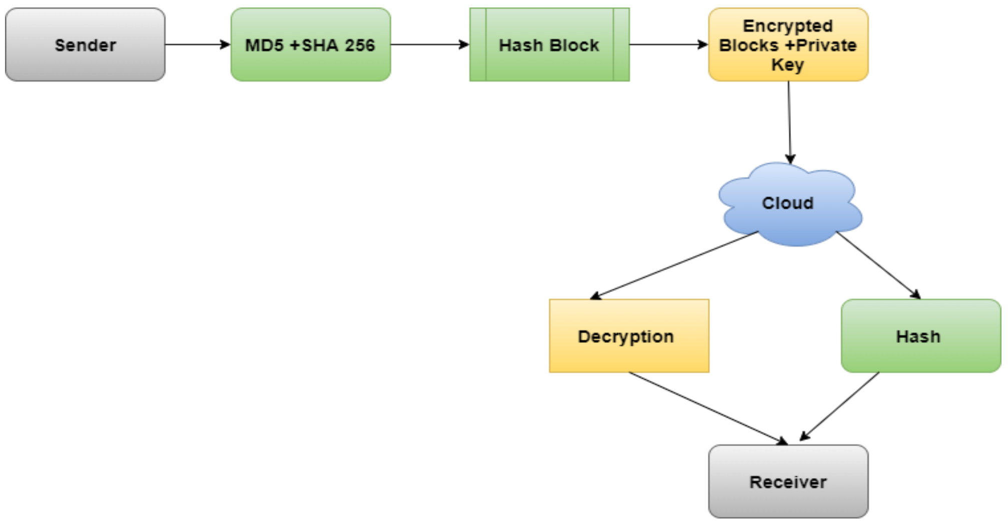

Figure 2: Block Diagram for Integrated Approach

3.2 M-Tree Hashing Technique

For computing and storage and verification, a sort of data structure called a hash tree or merkle tree is used. The offspring nodes of each non-leaf node in the tree structure are hashed together. Every child node is equally deep and is positioned as far left as it is possible to be. Hash functions are used to maintain the data's integrity. To demonstrate that a leaf node is a part of a particular binary hash tree, one must figure out how many of hashes proportional to the log of the total number of leaf nodes in the tree. In contrast, a hash list's amount is inversely proportional to the number of actual leaf nodes. A Merkle tree is a useful example of a cryptographic engagement method since the root of the tree is seen as a commitment while leaf nodes can be revealed and demonstrated to be a part of the original commitment[14][15].

A Merkle tree works by recursively hashing pairs of data together until a single hash value, called the root hash, is assembled. The root of the hash represents the complete dataset, and any change to the dataset will result in a different root hash value. The root hash is typically stored or transmitted with the dataset and can be used to verify the integrity of the dataset. To build a Merkle tree, the data is first split into fixed-size blocks, and each block is hashed using a cryptographic hash function such as SHA-256. The resulting hash values are then paired up and hashed together, and the process is repeated until a single root hash value is produced. To verify the integrity of a particular block of data in the Merkle tree[16], a client can request the hashes of the other blocks in the same level of the tree, along with the root hash. The client can then verify the authenticity of the data by hashing the requested block along with the other block hashes at the same level and comparing the resulting hash with the next level's hash value. This process continues until the client reaches the root hash, which can be compared with the transmitted root hash value. Merkle trees are used in many applications such as blockchain, distributed file systems, and peer-to-peer networks to verify the integrity of large datasets in a scalable and efficient manner. They are also used in digital signatures and secure communication protocols to verify the authenticity of messages and data.

Merkle trees are used in a variety of contexts in computer science and cryptography due to their ability to efficiently and securely verify the integrity of large datasets[17]. Here are some common use cases and needs for Merkle trees. Blockchain technology: One of the most well-known applications of Merkle trees is in blockchain technology. A Merkle tree is used in a blockchain to store all the transactions in a block. By hashing all the transactions together and building a Merkle tree, it's possible to create a compact and secure representation of the transactions that can be verified by all participants in the network. Distributed file systems: Merkle trees are often used in distributed file systems such as IPFS (Inter Planetary File System)[18] to verify the integrity of files that are distributed across multiple nodes. By building a Merkle tree of the file's content, each node can verify that the file has not been tampered with or corrupted by comparing the root hash of the Merkle tree with the hash value of the received data. Peer-to-peer networks: Merkle trees can also be used in peer-to-peer networks to verify the integrity of data being shared between nodes. By constructing a Merkle tree of the data, each node can verify that the data is correct and has not been corrupted or modified during transmission. Digital signatures Merkle trees are used in digital signature schemes such as the Merkle signature method, which allows several parties to jointly scribble a message. By building a Merkle tree of the message and the parties' signatures, a compact and secure representation of the signature can be produced that can be verified by any party in the network. Overall, Merkle trees are a powerful and versatile tool for verifying the integrity of large datasets and are used in many different applications in computer science and cryptography.

3.2.1 Steps to generate M-Trees:

When it comes to storage and processing speed, merkle trees are effective data structures. Data hash values that take up less space are stored there. Instead of searching the full tree, one can quickly scan for data by creating the Merkle path. The choice of hash functions affects the Merkle tree's effectiveness as well. The procedures the model use to construct a Markle tree are listed below:

1. Identify each data item's hash value. The child nodes of the Merkle Tree will be composed of these hash values.
2. Beginning from left to right, pair up the leaf nodes.
3. For each pair, make a parent node. Apply the hashing algorithm[19] to the aggregated value after adding the leaf nodes' hash values. This will serve as the parent node's label.
4. The parent nodes are then paired off, and the previous phase is repeated.
5. The Merkle root, or root hash, is the final node that remains after all the pairings have been completed.

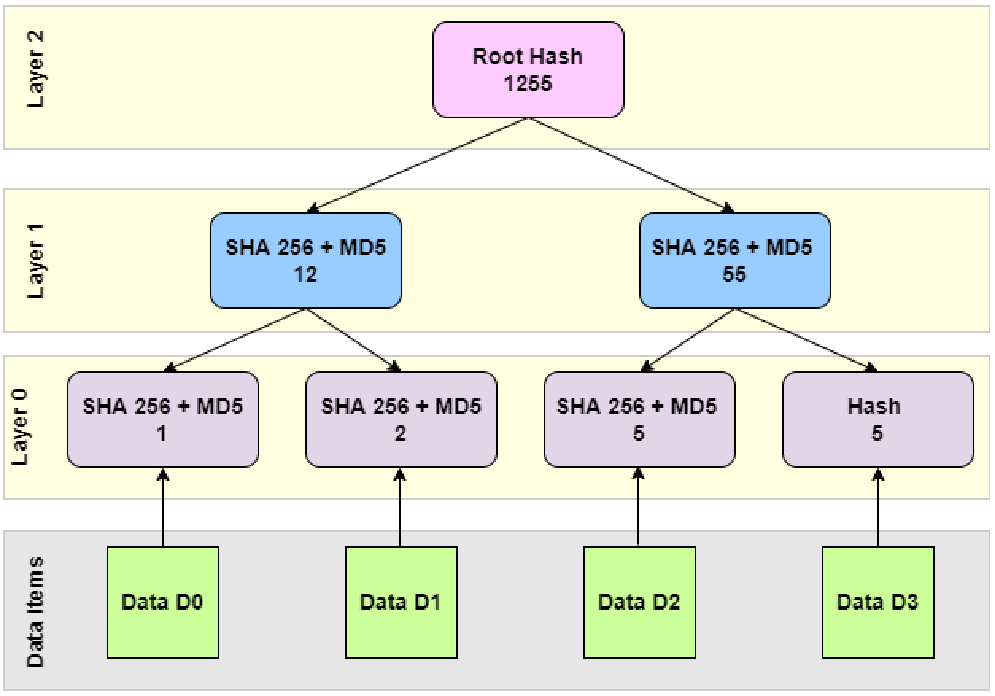

Figure 3: Block Diagram for Enhanced Hashing for Securing the Data

3.2.2 MD5 Algorithm

MD5 (Message Digest 5) operates by dividing the input message into blocks of 512 bits and processing each block through a series of transformations. These transformations involve bitwise logical operations, modular arithmetic, and logical functions such as AND, OR, and XOR. MD5 is widely used for generating digital signatures and message digests, but it has been found to be vulnerable to various attacks that allow an attacker to produce two different messages with the same MD5[20] hash value, also known as a collision. Due to these vulnerabilities, MD5 is no longer considered a secure cryptographic hash function for many applications. In many cases, cryptographic hash functions such as SHA-256 and SHA-3 are preferred over MD5 for their stronger security properties and resistance to known attacks. However, MD5 is still used in some legacy systems and applications that have not yet migrated to newer, more secure hash functions.

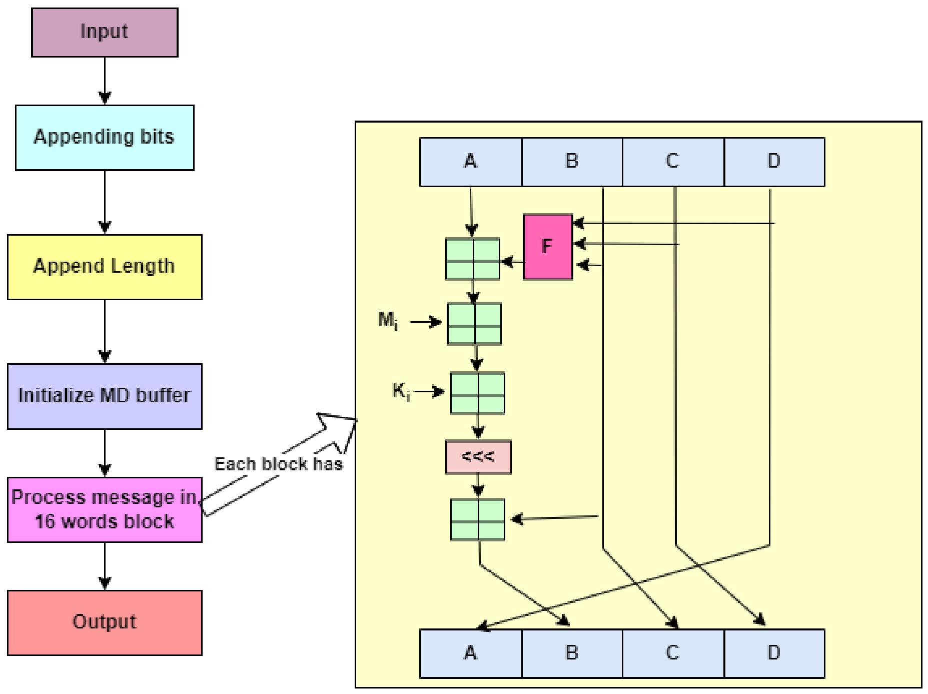

Figure 4: Process diagram of MD5

3.2.3 SHA-256 Algorithm

SHA-256 is widely used in various security applications to generate fixed-length, unique digital fingerprints of data. Enter data can be any length and provides a 256-bit (32-byte) output, which is commonly represented as a hexadecimal string of 64 characters. SHA-256 operates by breaking the input message into smaller blocks and processing each block using a series of mathematical operations. These operations involve shifting, rotating, XOR-ing, and modular arithmetic operations, which are performed on the input message and an initial set of constants to generate a set of intermediate values. These intermediate values are then combined and processed further to produce the final output. SHA-256[21][22] is considered to be a secure and widely used hash function and is utilized in many applications where digitalized signatures, authentication of messages, and password storage. It is also widely used in cryptocurrency applications such as Bitcoin and other blockchain-based systems.

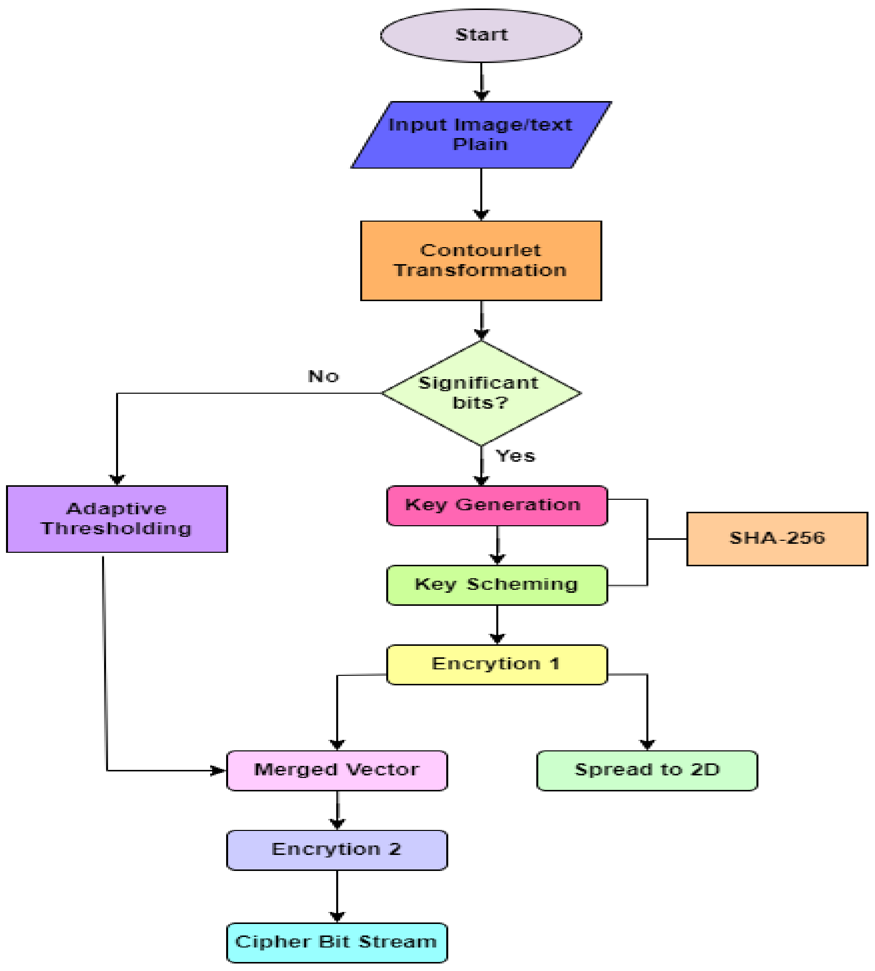

Figure 5: Flow Diagram for SHA-256

4. RESULTS & DISCUSSION

This section discusses about the various metrics associated with the cloud computing resources like encryption time, decryption time in milli seconds, response and execution time. The memory utilization is represented in MB. The experiment is executed on the cloud sim environment, which is an open source simulation tool. This tool helps the entities to communicate with the standard policies. The latest version i.e., 5.0 is used in this proposed research. Table 2 presents the time to encrypt the files. The encryption time is less in the proposed model when compared to solo model.

Table 2: Time to Encrypt the file blocks

Input Size in MB	**MD5**	**SHA-256**	**M-Trees**	**Integrated M-Tree**
1000	0.81	0.72	0.62	0.59
1250	0.88	0.84	0.73	0.65
1500	0.91	0.87	0.85	0.79
1750	0.97	0.94	0.91	0.83
2000	0.99	0.96	0.95	0.90

Table 3 presents the decryption time on different blocks of data. It is observed that with the increase in block size the decryption time also increases. But the decryption time in the integrated approach is less when compared to the traditional approaches

Table 3: Time to decrypt the file blocks

Input Size in MB	**MD5**	**SHA-256**	**M-Trees**	**Integrated M-Tree**
1000	0.87	0.71	0.65	0.62
1250	0.89	0.88	0.79	0.68
1500	0.93	0.90	0.87	0.73
1750	0.99	0.96	0.94	0.85
2000	1.20	1.01	0.98	0.91

Figure 6 represents the overall execution and response time in milliseconds consumed by various methods, which are contrasted with the proposed method. The integration of MD5 and SHA-256 leads to good outcomes. It can be observed from the table that Integrated M-Trees have given quick responses than the others. IMTree takes less than 3 minutes to execute and responds in less than 1.5 minutes. If observed, the SHA-256 takes nearly 4 minutes for execution which is quite higher than any other method compared. The late response is also given by it, which is more than 2 minutes.

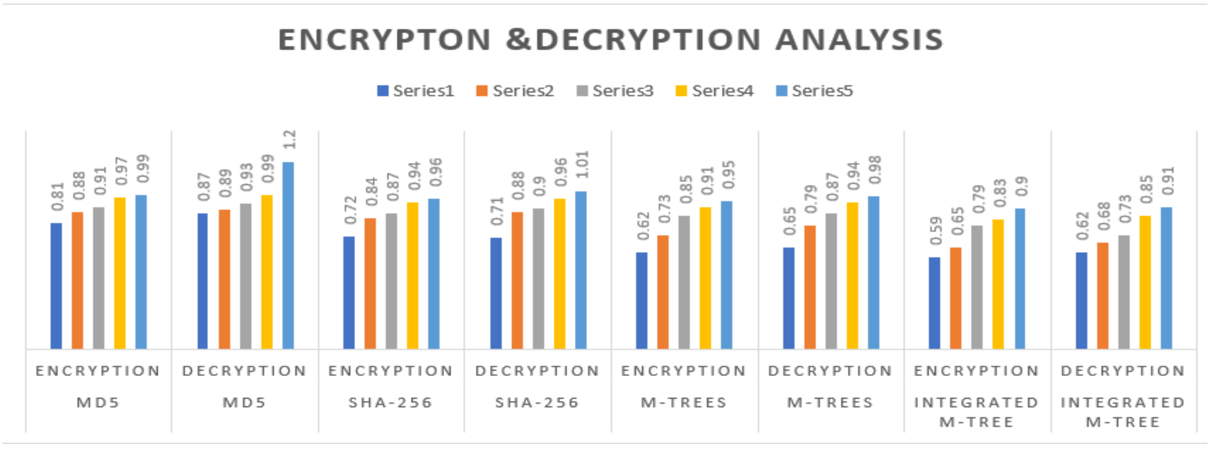

Figure 6: Encryption & Decryption Analysis

Table 4 compares the execution time & response time of various algorithms in terms of milli seconds and it is observed that execution and response time is drastically changed in proposed model.

Table 4: Execution & Response Time Comparison

Algorithm	Execution Time	Response Time
MD5	208817	100222
SHA-256	247721	128024
M-Trees	204174	108821
IMTrees	174188	88244

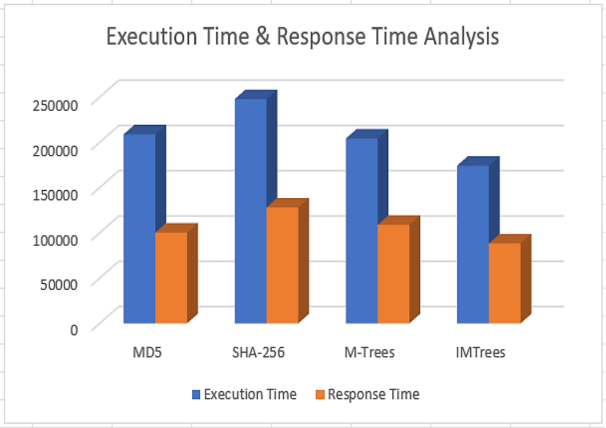

Figure 7: Analysis on Execution & Response Time

Figure 7 is the depiction of the time taken for execution and response by various methods respectively. The time taken for execution by each method is represented with a blue color bar and the red color bar is the time taken for the response. From the bar chart, the model can come to a note that the IMTrees can exceptionally reduce the duration.

Table 4: Memory Utilization Analysis

S.No	Algorithm	Memory Utilization in MB
1	**MD5**	190
2	**SHA-256**	125
3	**M-Trees**	118
4	**IMTrees**	78

Table 5 represents the memory utilization of the methods MD5, SHA-256, M-Trees, and the suggested IMTrees. It is seen that the integrated M-Trees consumes just 78MB. This is less than the 50% consumption of memory by MD5. MD5 takes a high amount of memory to run or execute. If the memory consumption is high, it will thaw out.

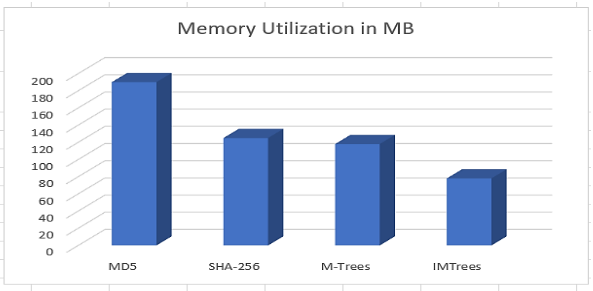

Figure 8: Memory Utilization Analysis

Figure 8 shows the analysis of the utilization of memory to work. The amount of memory typically used is determined by the percentage of total memory available at any given time. In the postulated method, IMTree used less memory. The memory utilization is shown in MegaBytes. Optimal utilization of memory will impact in the cost, power consumption, etc.

5. CONCLUSION

The TPA must spend a lot of time auditing each encrypted block, which raises the deployment cost. Cloud resources employ either the rejuvenation code each time it is shared with others or the proxy server to ensure source access management. The "SSL stripping" that affects proxy servers occurs when an attacker sends the https URL and steals the victim's information. The suggested research integrated the SHA-256 and MD5 algorithms because, when information is being transferred, communications need to be quick, which MD5 provides, and security needs to be maintained, i.e., data needs to be encrypted in a compact and secure format, which SHA-256 provides. When utilising SHA-256, the input message is divided into smaller blocks, and each block is subjected to several mathematical processes. For developing virtual signatures and message digests, MD5 is frequently employed. The results are clear that the proposed algorithm integrated M-tree has shown excellent results. The time taken to encrypt or decrypt are notably lesser than the compared M-tree or MD5 etc. As the file size varies, encrypting time is relatively proportional.

REFERENCES

[1] Ravikumar, G. ., Begum, Z. ., Kumar, A. S. ., Kiranmai, V., Bhavsingh, M., & Kumar, O. K. . (2022). Cloud Host Selection using Iterative Particle-Swarm Optimization for Dynamic Container Consolidation. International Journal on Recent and Innovation Trends in Computing and Communication, 10(1s), 247–253. https://doi.org/10.17762/ijritcc.v10i1s.5846

[2] K., V. R. ., Yadav G., H. K. ., Basha P., H. ., Sambasivarao, L. V. ., Rao Y. V., Balarama K. ., & Bhavsingh , M. . (2023). Secure and Efficient Energy Trading using Homomorphic Encryption on the Green Trade Platform. International Journal of Intelligent Systems and Applications in Engineering, 12(1s), 345–360. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/3421

[3] Liu, J., Huang, K., Rong, H., Wang, H., & Xian, M. (2014). Privacy-Preserving Public Auditing for Regenerating-Code-Based Cloud Storage. IEEE Transactions on Information Forensics and Security, 10(7), 1412–1428. https://doi.org/10.1108/TIFS.2014.2416688

[4] Kumar, R., Nagar, G., Rajit, I., & Assistant, N. (2016). Multi-Cryptosystem based Privacy-Preserving Public Auditing for Regenerating Code based Cloud Storage. In International Journal of Computer Applications (Vol. 144, Issue 10).

[5] Liu, M.-P., Jiang, R., & Kong, H.-F. (2017). Cryptanalysis and Countermeasures on Privacy-Preserving Public Auditing for Regenerating-Code-Based Cloud Storage. http://creativecommons.org/licenses/by-nc/4.0/

[6] Satya, A. L., Raju, I. R. K., Bhaskara Murthy, S. v, & Student, M. (n.d.). Privacy-Preserving Public Auditing For Regenerating-Code-Based Cloud Storage. www.jespublication.com

[7] Anantharam, B., Sharma, N. ., & Rani, B. . (2023). Implementation of Dynamic Virtual Cloud Architecture for Privacy Data Storage. *International Journal on Recent and Innovation Trends in Computing and Communication*, *11*(11s), 177–184. https://doi.org/10.17762/ijritcc.v11i11s.8084

[8] Nalini, S. P., & Ponmagal, R. S. (2016). SECURE PRESERVING PUBLIC AUDITING FOR REGENERATING CODE BASED ON CLOUD STORAGE. In International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) (Vol. 21).

[9] Yu, Y., Au, M. H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., & Min, G. (2016). Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Transactions on Information Forensics and Security, 12(4), 767-778.

[10] Vincent, T., & Krishnaveni, M. V. V. (N.D.). Ijesrt International Journal Of Engineering Sciences & Research Technology Robust And Efficient Privacy Preserving Public Auditing For Regenerating-Code-Based Cloud Storage. © International Journal of Engineering Sciences & Research Technology. Https://Doi.Org/10.4281/Zenodo.802828

[11] Thokchom, S., & Kr Saikia, D. (2018). Privacy Preserving and Public Auditable Integrity Checking on Dynamic Cloud Data. International Journal of Network Security, 21(2), 221–228. https://doi.org/10.6622/IJNS.201802

[12]Tang, X., Huang, Y., Chang, C. C., & Zhou, L. (2018). Efficient Real-Time Integrity Auditing with Privacy-Preserving Arbitration for Images in Cloud Storage System. IEEE Access, 7, 22008–22022. https://doi.org/10.1108/ACCESS.2018.2804040

[13] Nayomi, B. D. D. ., Mallika, S. S. ., T., S. ., G., J. ., Laxmikanth, P. ., & Bhavsingh, M. . (2023). A Cloud-Assisted Framework Utilizing Blockchain, Machine Learning, and Artificial Intelligence to Countermeasure Phishing Attacks in Smart Cities. International Journal of Intelligent Systems and Applications in Engineering, 12(1s), 313–327. Retrieved from https://ijisae.org/index.php/IJISAE/article/view/3419

[14] G, P., Dunna, N. R., & Kaipa , C. S. (2023). Enhancing Cloud-Based IoT Security: Integrating AI and Cyber security Measures. International Journal of Computer Engineering in Research Trends, 10(5), 26–32. https://doi.org/10.22362/ijcert.v10i5.43

[15] Lakshmi, M. S. ., Ramana, K. S. ., Pasha, M. J. ., Lakshmi, K. ., Parashuram, N. ., & Bhavsingh, M. . (2022). Minimizing the Localization Error in Wireless Sensor Networks Using Multi-Objective Optimization Techniques. International Journal on Recent and Innovation Trends in Computing and Communication, 10(2s), 306–312. https://doi.org/10.17762/ijritcc.v10i2s.5948

[16] Abou Bakary Ballo, & Diarra Mamadou. (2023). A Comprehensive Study of IoT Security Issues and Protocols . International Journal of Computer

Engineering in Research Trends, 10(7), 8–14. https://doi.org/10.22362/ijcert.v10i7.858

[17] Md. Saad Amin, Primitiva Morales-Romero, Miguel Chamorro-Atalaya, & M.Bhavsingh. (2023). A Novel User Interface Design for Enhancing Accessibility in Mobile Applications. International Journal of Computer Engineering in Research Trends, 10(8), 26–33. https://doi.org/10.22362/ijcert.v10i8.873

[18] Guillermo Ramos-Salazar, Sabrina Rahaman, & Md. Amzad. (2023). A Machine Learning-based Approach for Detecting Malicious Activities in Cloud Computing Environments. International Journal of Computer Engineering in Research Trends, 10(9), 22–28. https://doi.org/10.22362/ijcert.v10i9.872

[19] MAHDI, S. J. (2016). Preventing From Collusion Data Sharing Mechanism for Dynamic Group in the Cloud. International Journal of advanced Research in Computer Science, 2(7), 113-118.

[20] Mallesh Goud, Pratika Malya.(2015). Dynamic Group data sharing framework on Cloud Servers. Macaw International Journal of Advanced Research in Computer Science and Engineering.1(1),16-20.

[21] Lalchand G. Titare , Riya Qureshi (2016). Cloud based IoT for Agriculture in India. Macaw International Journal of Advanced Research in Computer Science and Engineering.2(12),5-10.

[22] Pradeep G, P Satyanaryana, & S. Ramamoorthy. (2023). A Novel Cryptographic Protocol for Secure Data Sharing in Cloud Computing . *International Journal of Computer Engineering in Research Trends*, *10*(11), 10–18. https://doi.org/10.22362/ijcert.v10i11.884