Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

Secure cloud data storage solution with better data accessibility and time efficiency

K. Roslin Dayana & P. Shobha Rani

To cite this article: K. Roslin Dayana & P. Shobha Rani (2023) Secure cloud data storage solution with better data accessibility and time efficiency, Automatika, 64:4, 751-758, DOI: 10.1080/00051144.2023.2213564

To link to this article: https://doi.org/10.1080/00051144.2023.2213564

Published online: 25 May 2023.

AUTOMATIKA
2023, VOL. 64, NO. 4, 751–758
https://doi.org/10.1080/00051144.2023.2213564

Secure cloud data storage solution with better data accessibility and time efficiency

K. Roslin Dayana and P. Shobha Rani

Department of Computer Science and Engineering, R.M.D. Engineering College, Kavaraipettai, Tiruvallur, India

ABSTRACT

The environment of cloud computing provides several advantages and a variety of data storage models, which entirely frees users from the vexing processes of storage equipment upgradation and data administration. Nevertheless, the customers' primary worry is the safety of their data, and the literature offers a number of different security-based solutions for addressing this issue. This paper proposes a technique for time-efficient cloud data storage that makes use of keccak and Advanced Encryption Standard (AES) which enhances the data availability with a reasonable storage space. Third party agents are not employed, due to the fact that a third party cannot be relied upon to maintain a high level of data security. Thus, the proposed algorithm is applied by cloud users before the process of outsourcing is carried out. This method is resistant to data tamper and analytical attacks. In addition, the execution of the work that has been proposed requires a limited amount of time, which, in turn, minimizes the amount of energy that is required. In contrast to the already used algorithms, the presented work demonstrates superior performance in terms of its overall effectiveness.

ARTICLE HISTORY
Received 20 February 2023
Accepted 8 May 2023

KEYWORDS
Cloud data accessibility; time efficiency; energy conservation; security; data outsourcing

1. Introduction

In today's environment, each and every second is spent dealing with a vast amount of data. It is asserted that most of the data are secret since they contain information that is personally identifiable as well as health and economic details. The quantity of data continues to grow over time, making it extremely challenging for enterprises to both store and manage the information. In addition to this, the firms are required to acquire costly storage equipment in order to function properly. Another significant problem that the firms have to deal with is the handling of their data.

The idea of computing in the cloud is a boon to enterprises of both moderate and small sizes since it provides the greatest option to avoid setting aside a significant amount of money for the purposes of data storage. In addition, cloud computing gives users access to a plethora of appealing qualities, including on-demand services, simple and affordable payment methods, simplified data upkeep, and many more. These characteristics encourage data owners to store their data on the cloud and outsource the management of their data.

In this manner, the owners of the data are able to enjoy minimized overhead costs concerning data management and storage [1–5]. Yet, the data owners are hesitant to hand over their sensitive data to an untrustworthy Cloud Service Provider (CSP) [6–10]. This raises a number of concerns relating to security, including those pertaining to the availability, integrity, and confidentiality of data. For example, certain clouds have behaved inappropriately by erasing or otherwise altering data that is only seldom accessed [11,12].

It has been stated that Amazon S3, Gmail, and Sidekick all suffer from the same kind of problem, which manifests itself as a failure, the deletion of emails, and a disaster accordingly [13]. As a result, it is necessary to protect the accuracy of the data, and the data's integrity must be preserved at all times. This research article's primary objective is to propose a secure data storage model that maintains the data integrity of cloud storage while ensuring better data accessibility. This is accomplished by combining two separate security protocols.

The work that is being proposed ensures the data integrity of the information that is saved in the cloud by utilizing the keccak hash function and the Advanced Encryption Standard (AES) for the purpose of carrying out the encryption operation. Therefore, the work that is being presented offers dual layers of protection. The efficacy of the proposed work is evaluated in comparison to the SHA-3 member algorithms, including BLAKE, SKEIN, and SHA2, with respect to throughput and memory requirements. In addition to this, the amount of time required for encryption is also indicated below. The key points of this article are summarized in the following list.

CONTACT K. Roslin Dayana krd.cse@rmd.ac.in Department of Computer Science and Engineering, R.M.D. Engineering College, Kavaraipettai, Tiruvallur, India

- A security mechanism for cloud storage that employs both hashing and encryption is described above as having two layers of protection.
- Because the data nodes are organized in a structured tree style, it is possible to update any section of the data at any given moment in time.
- There is no need to look up any tables, and there are no tables that are maintained.
- Hashing process of this work relies on Keccak's hash function, which enhances the security and processing speed. This results in a reduction in the amount of memory that is required, which in turn enhances performance.
- The task that is being proposed does not include any arithmetic or rotational operations.
- The AES algorithm is used for encryption because it is well-known for being both quick and effective.

The remaining portions of the article are organized in the following fashion: In the following section, the relevant literature with regard to maintaining secure cloud storage is examined. The proposed method is broken down in greater detail in section 3. In section 4, the effectiveness of the proposed work is proven. Section 5 concludes the article.

2. Review of literature

This section reviews the related literature concerning data outsourcing schemes of cloud computing.

In [14], an identity-based Provable Data Possession (PDP) scheme is presented on the basis of RSA assumption for privacy ensured secure cloud storage. The identity-based homomorphic authenticators are produced by taking the outsourced file and a time-bound global parameter. The integrity verification of this work is proven to be better with reduced computational and communicational overhead.

A Lattice-based data outsourcing scheme is presented with public integrity verification in [15]. The data integrity is verified by Third Party Auditor (TPA) and this work ensures that the data is not deleted without the concern of TPA. In [16], a secure identity-based data outsourcing scheme is presented for public integrity verification in cloud storage. This scheme is based on lattice-based cryptography that prompts the data owner to produce the signature of the data before outsourcing. The data integrity is verified by the TPA, without accessing the complete data. The certificate management processes are also not required.

A range query scheme for outsourced data for the cloud is presented in [17]. This work utilizes pivot mapping for data mapping to a low-dimensional space. In addition, the data is encoded and the maps are coded to multiple bloom filters, which are organized in a binary tree-based index. This work ensures data privacy and the data similarity is hidden. In [18], an identity-based signcryption scheme is presented with revocation. This work focuses on confidentiality, integrity, and authentication too.

A revocable attribute-based data storage scheme for mobile clouds is presented in [19]. This work does not prompt the data owners to show the authorized users and the mobile users can authorize the Cloud Service Provider (CSP). Finally, the file re-encryption process and access update are carried out offload without disturbing the users. In [20], a system with provenance is presented for outsourced data with the blockchain approach. The access logs are secured by blockchain while maintaining two categories of data users and accounts.

In [21], a cloud access control framework is presented on the basis of intelligent big data analytics. This work utilizes a global identifier for providing access control while providing data acquisition and decentralized audit and ordering chain mechanisms. A Provable Data Possession (PDP) scheme is presented with outsourced data transfer in [22]. This work focuses to verify the security of the data belonging to an acquired enterprise, to verify the integrity and privacy of the data, and to outsource the transferable data computation. In [23], a secure outsourcing scheme is presented on the basis of user-side encrypted file system. This work presents a user-based encryption technique, which is a hybrid of symmetric and asymmetric techniques with Identity Based Encryption (IBE).

In [24], a cloud storage system is presented on the basis of batch leaves authenticated merkle hash tree. This work supports dynamic updates to the outsourced data and the indexes of all the leaf nodes are maintained. A highly secure k-Nearest Neighbour (k-NN) classification technique is presented for cloud data in [25]. This work relies on homomorphic cryptosystem and secret sharing, which employs secure comparison, sorting, frequency calculation and so on for attaining efficiency.

A privacy preserving Support Vector Machine (SVM) classification based data outsourcing is presented in [26]. Here, a client-server data classification protocol is presented on the basis of SVM classifier. This work can handle privacy preservation problems for two and multiple classes. Secure two-party computation and pailler homomorphic encryption are incorporated in this work.

Inspired by these works, this article aims to present a secure cloud storage solution that is based on Keccak hash tree and AES algorithm, which can assure better data accessibility.

3. Proposed secure cloud data storage solution with effective data accessibility

Despite the fact that there are a number of research problems linked with cloud computing, security has

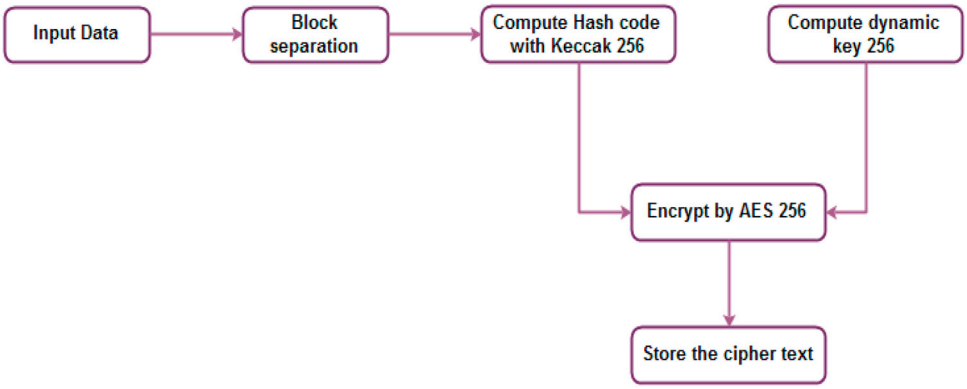

Figure 1. Overview of the proposed work.

been the most researchable topic ever since its introduction. Cloud services offer several benefits, including elasticity, scalability, flexibility, and pay only for the services that are utilized, and so on, which is why they are appealing to businesses of all sizes, particularly those of medium and small scale. Users have a great deal of mistrust or scepticism in their CSPs, due to the unpleasant experiences they have had with data exploitation or leaking in real-world situations. The primary concern of cloud users is the safety of their data [27–33]. Figure 1 illustrates the process through which the work is completed in its entirety.

As a result, CSPs cannot be relied upon entirely by customers, and users themselves need to take certain safeguards. Encryption, which transforms the data into a format that cannot be understood, is one of the safety measures that has been confirmed to be the most successful in preventing the misuse of data. The CSP is only able to access the data after first conducting a number of manipulations on them, and the ease with which the data may be accessed depends on how well the encryption technique works. When an encryption technique is more effective, it becomes increasingly difficult to penetrate the security line and interpret the data. The keccak method uses a tree data structure and is based on the idea of hashing, which is a cryptographic function. This assists in achieving good data arrangement and helps to achieve better data accessibility. It also contributes to the effectiveness of the overall system.

The information is arranged in a bottom-up fashion, and the root node contains the hash code for the entire tree structure. As a result of the fact that the root node is the most significant component of the tree, the AES algorithm is utilized once more in order to encrypt the hash code of the root node. This concept ensures a higher level of security and has the potential to successfully defend against assaults including data analysis and data tampering. It is quite tough to manipulate the data or analyze the data due to the fact that this task relies on hash code. The standard organization of data also makes it very difficult to analyze the data. The next part provides an explanation for each stage that will be involved in carrying out the proposed work.

3.1. Keccak with AES

As was mentioned previously, the Keccak hash function is a strong candidate for the SHA-3 algorithms, and it is founded on the concept of sponge formation. The creation of the sponge is determined by the relationship between two significant parameters called bit rate (br) and capacity (c). The sponge is formed based on the values of br and c, and the total of these two values determines the permutation of the breadth of the keccak. Initialization, absorption, and squeezing are the three crucial steps that are involved in the operation of the keccak algorithm, which can be seen as its underlying working principle. Figure 2 illustrates the basic operations that are performed by the algorithm.

The keccak hash function is employed after the input data has been segmented into a large number of smaller data chunks during the phase of startup. During the second phase, the XOR operation is conducted with the current matrix using the data chunks that have fixed standard bits and 24 rounds of permutations are performed. After all of the input data have been processed in this manner, the squeezing phase begins. During this phase, the final matrix will be truncated, in order to obtain the necessary hash value.

Consider the event that when the minimum needed hash length is not attained, the procedure is redone until the minimum required hash length is attained. These steps are necessary for the development of sponges, and they can be summed up below. The sponge creation of the keccak hash tree operates on bits, and it adheres to the idea that the total of *br* and *c* must be as high as possible, which is currently 1600. This value has been permanently set to prevent bit attacks, which is the reason for it. The work that is being proposed involves 256 bits length for the output. The data to be outsourced is represented in the following way.

$$Data = \{Data_{bt1}, Data_{bt2}, Data_{bt3}, \ldots, Data_{btn}\}$$

All of Keccak's operations are done using byte-based operations. Now that all of these separated blocks have been stacked in tree fashion, the construction of the sponge is complete. After that, each bit is placed in its

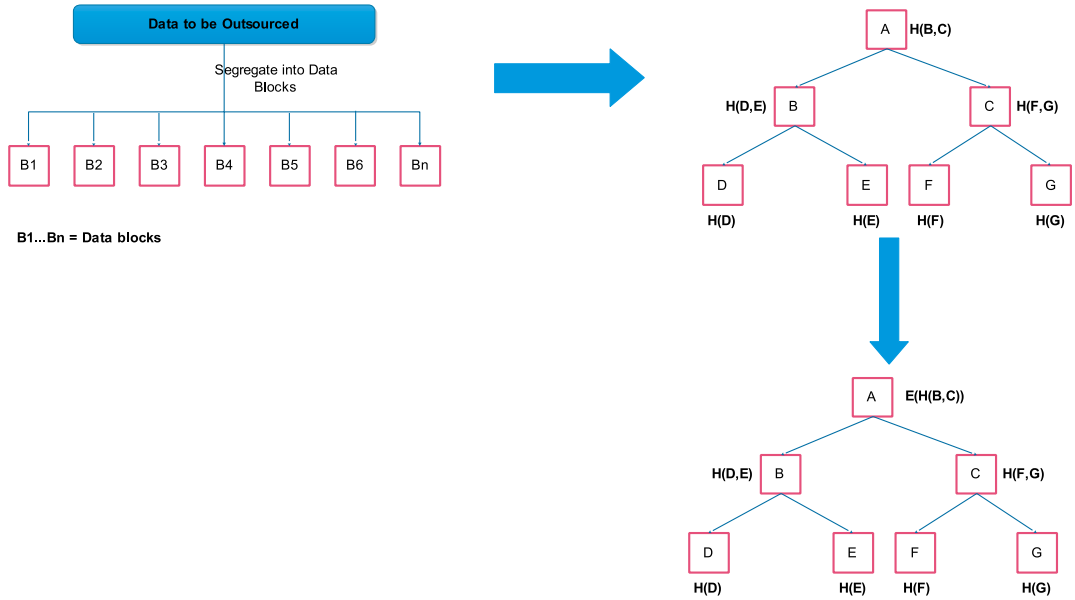

Figure 2. Proposed secure cloud data storage with better data accessibility.

proper location at the base of the tree, and the nodes on the level below that one are responsible for calculating the hash value. The keccak hash algorithm is utilized in the computation of the hash value, and the structure of the hash tree is laid out as follows.

In this case, the hash code of all of the data that is present in the tree is stored at the node that is considered to be the tree's root. With the assistance of the AES algorithm, this work encrypts only the root node of the tree rather than encrypting all of the nodes that are engaged in the tree.

Keccak Sponge Formation

Input : Data CD;
Output : Formed sponge;
Begin
ST[a,b]=0 for all (a,b) in 0 to 4;
BL = Divide CD into bytes;
// Absorption
For all BL_i *in BL*
Do
$ST\ [a,b]=ST[a,b] \oplus BL\ [a+5b]$
$ST = keccak - f[br + c](ST)$
End do
End for;
//Squeeze
Q = Empty String;
Q=Q || ST[a,b] //String concatenation
$ST=Keccak - f\ [br + c](ST)$
Return Q;
End;

Following the establishment of a tree with hash codes, the implementation of encryption contributes an additional layer of protection to the overall framework of the system. During this stage of the process, the well-known encryption technique AES is used. The execution speed of AES is well-known, as is the high level of security it provides. The hash codes are encrypted using the dynamic key so that the level of security can be increased. The dynamic key is not divulged to any third party by the owner of the data at any time. The CodeIgniter key generator, which has a length of 256 bits, is the one responsible for producing the dynamic key for this work.

Because it operates across bytes rather than bits, AES is the encryption method of choice for this activity. In this study, the Advanced Encryption Standard (AES) is utilized with a key size of 256 bits, which results in 14 rounds of encryption. The level of system security also improves along with the amount of rounds that are played in a game. Every single cycle makes use of a 128-bit key, the computation for which is performed by the AES key itself. This section presents the algorithm used for the encryption step.

Encryption phase

Begin
For all data blocks
Do
Compute dynamic key 'd' by CodeIgniter key generator;
Pass d and hash code to AES;
Return cipher text;
End;
End;

Every round of AES requires a number of significant actions, including shifting rows and columns, mixing columns, substituting bytes, and adding round keys. The bytes that are being input are initially passed into the S box, which is laid out in the matrix form. After that, the rows are rearranged taking into account the

positioning constraints and then the columns are mixed up by switching the old bytes with new ones, and this process stops in the 14th round by itself.

The XOR operation is performed with the round key when it comes to the final components of the matrix. In order to produce the cipher text, this procedure must be carried out for a total of 14 rounds. The security of the system is significantly bolstered by the fact that the dynamic key is never divulged to anyone. Additionally, the original data text is not directly encrypted; rather, the hash codes that are produced are encrypted.

Therefore, the results of this investigation give security on two different levels. When the owner of the data has to perform an integrity check, the AES algorithm is decrypted using the dynamic key that is kept a secret. This allows the hash code to be acquired. After that, the hash codes can be compared with one another in order to validate the data's authenticity. By posing a challenge to the cloud server, the owner of the data can run the integrity check whenever they want at any moment in time. The next section evaluates and discusses work performance.

4. Results and discussion

The effectiveness of the proposed work is evaluated by simulating it in Java on a stand-alone computer with 8 GB of RAM and one TB of hard drive space. The performance of the proposed work is evaluated from three different perspectives. Initially, the performance of the proposed work is analyzed with respect to the member algorithms of SHA3. The second round of performance analysis is done by contrasting the proposed work with Merkle hash tree. The third round of performance analysis is performed to justify the choice of AES algorithm.

The amount of time it takes to carry out an algorithm is yet another crucial indicator for measuring its performance. In order to reduce latency as much as feasible, the amount of time spent in execution needs to be kept to a minimum. It is not possible for an algorithm to have a good performance and a long execution time at the same time. Variations in the data size are used in an effort to determine how the suggested twin-layered security method would perform in terms of its execution time. This experiment is conducted in two distinct settings by making adjustments to the amount of output that is received. The length of the output can be somewhere between 256 and 512 bytes.

It is possible to deduce from graph 3 that the length of the output plays a significant role in determining the amount of time required for the execution. The Intel Core i7 processor was used to get the result that was just discussed. When the output length is fixed at 512 bytes, the process of execution takes up less time than when it is set to 256 bytes, which causes the execution process to take up more time overall. When speed is prioritized

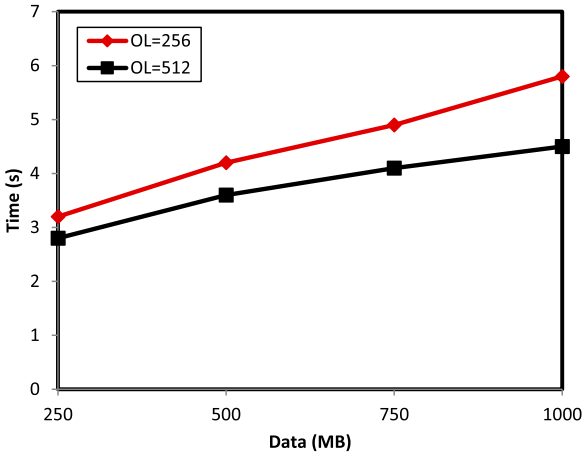

Figure 3. Execution time analysis.

over security, the output length can be lengthened to accommodate the higher pace. When the output length is set to 256, the br, c, and d values are respectively set to 1088, 512, and 32. When the output length is 512 bytes instead of 512 bytes, the values of br, c, and d are respectively set to 576, 1024, and 64. This is another option. Therefore, an increase in the total number of rounds results in an improvement to the system's security Figure 3.

Thus, the main objective of the work is to present a cloud data security model with better data accessibility is achieved. The proposed security model is examined with several performance metrics and the proposed work outperforms the comparative algorithms.

4.1. Performance comparison between Keccak and Merkle Hash tree

The existing auditing mechanisms confront multiple drawbacks. Some of them are communication overhead, authentication issues. These issues are proposed to be addressed by the proposed work by employing Keccak's hash tree. The public auditing mechanism to be proposed considers the degree of nodes, such that the verification process of multiple replicas becomes efficient and effective.

Keccak's hash tree processes each block of the data and it can process multiple blocks of data at a time. As parallel processing is made possible, the execution time is very much lower. The parallel computation can be classified on the basis of multiple or single message.

In the case of multiple message scheme, the data blocks which are sponges are not dependent on each other. If it is a single message type, then the sponges are interdependent and hence the outcome of a sponge will be the input of another sponge.

The number of nodes in a keccak's tree can be decided by considering the length of the data. Every node in the tree processes the sponge and the higher-level node depends on the lower-level node. The outcome is returned in the root node of the tree.

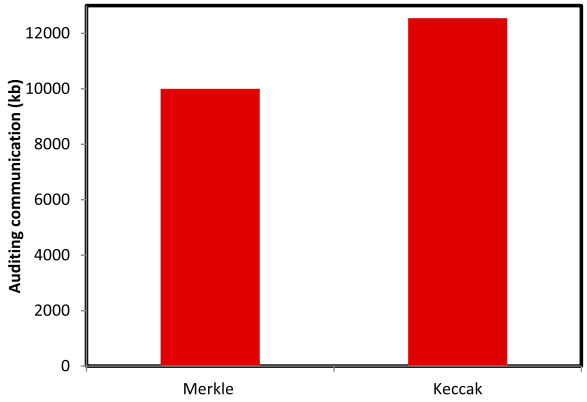

Figure 4. Auditing communication analysis.

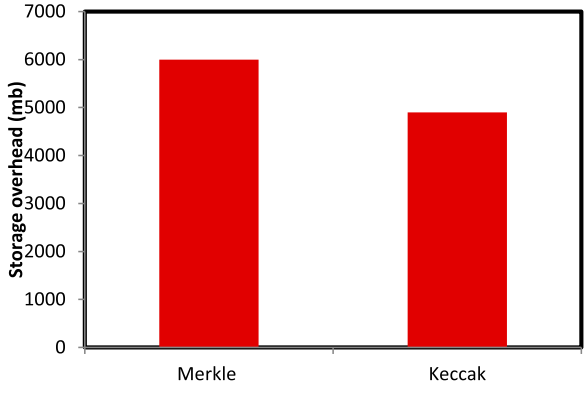

Figure 5. Storage overhead analysis.

The inputs passed to this tree are height, degree, block size to be processed, rate, and capacity. The parameter rate decides the speed at which the data has to be absorbed by the node. Capacity is the number of bits. Height is the number of levels that a tree can grow. Degree is the parameter that decides the number of children that a node can possess.

The growth mode of the keccak's tree to be used in our work is leaf interleaving, as such a tree can work for any sized input. In addition to this, the proposed work compares the performance of the Keccak is analyzed against Merkle hash tree in terms of auditing communication, storage overhead, and data transfer.

Auditing communication overhead is considered, as this work maintains a tree like structure with leaves. As the work has to consider all the data in the leaves, the auditing communication is a bit higher than the Merkle hash tree. On the other hand, the storage overhead is minimal for the proposed approach with a better data transfer rate, as shown in Figures 4–6. The proposed work is analyzed to prove the efficacy of AES algorithm in the following section.

4.2. Performance analysis by varying encryption algorithms

This section aims to emphasize the efficiency of the proposed security scheme in two ways. Initially, the choice of AES algorithm is proven by comparing with the analogous algorithms such as DES, Blowfish in terms of

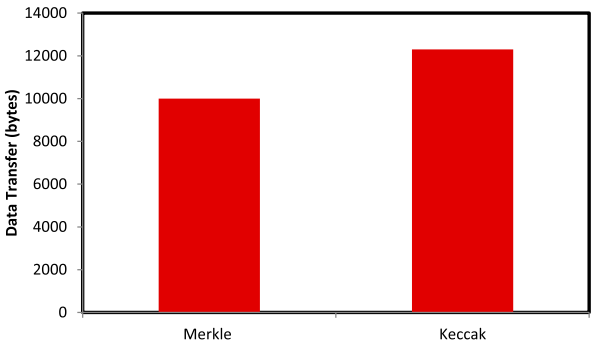

Figure 6. Data transfer rate analysis.

Table 1. AVS of different encryption algorithms.

Place of Bit modification	Blowfish	DES	AES
Initial	0.008	0.43	**0.996**
Middle	0.007	0.46	**0.994**
Last	0.007	0.44	**0.996**

encryption time and throughput. Finally, the performance of keccak and AES algorithm is analyzed by Avalanche effect.

In the proposed approach, AES-256 is utilized and the reason for the choice of AES-256 is justified by performing comparative analysis with the analogous encryption algorithms such as Data Encryption Standard (DES), Blowfish, AES-128. The efficiency of encryption can be judged with the help of Avalanche effect.

The avalanche effect is measured by the variation of output data when the input data is altered. Hence, an encryption algorithm must show greater avalanche effect score, such that any alteration made can show significant changes. Suppose when a single bit is modified, considerable changes are observed. The avalanche score (AVS) ranges in the scale of 0–1 and is computed by the following formula.

$$AVS = \frac{C_{mb}}{T_b}$$

C_{mb} is the total count of modified bits in the cipher text and T_b is the total number of bits in the cipher text. When the AVS is greater, the encryption algorithm is proven to be stronger and more efficient. The following Table 1 presents the AVS of different encryption algorithms. The cipher text is altered by a single bit, to carry out this analysis in different places (initial, middle, and last).

This analysis is carried out by varying the size of the data and the average AVS is presented in the above table. On observing the AVS, the AES algorithm is found to be a better performer than other algorithms and so AES is chosen for TSS. Additionally, the time consumption of AES is analyzed and the results are as follows. This analysis is performed by varying the input size Table 2.

The time complexity is measured in terms of milliseconds. Blowfish algorithm consumes minimal

Table 2. Time consumption analysis of encryption algorithms.

Size of input (kb)	Blowfish (ms)	DES (ms)	AES (ms)
10	4.8	8.3	**11.9**
39	8.8	33.2	**20.9**
56	16.20	53.6	**22.4**
112	26.8	67.4	**32.8**
168	36.7	83.1	**43.1**

Table 3. AVS of the combination of encryption algorithms.

Place of Bit modification	Blowfish + Keccak	DES + Keccak	AES + Keccak
Initial	0.032	0.48	**0.998**
Middle	0.029	0.49	**0.997**
Last	0.030	0.47	**0.998**

period of time, but it fails in exhibiting better AVS. The main intention of this work is to ensure better security rather than time conservation. Yet, the time consumption is tolerable. The time consumption of DES is greater, as the size of data increases. AES shows stable performance. For this reason, AES is utilized in combination with keccak. The AVS of the combination of keccak with other algorithms is presented in the following table Table 3.

Hence, the choice of AES among the encryption algorithms and the combination of keccak with AES is justified. The proposed approach is evaluated by varying the encryption algorithms, SHA3 member algorithms. Finally, the keccak is combined with different encryption algorithms and the results are discussed. From the experimental results, it is proven that the performance of the combination of AES with Keccak is proven with better results. The following section concludes the chapter.

5. Conclusions

This article describes a time-saving and safe cloud data storage solution with better data accessibility based on keccak hash tree and AES algorithms. However, without jeopardizing the algorithm's integrity, the purpose of this study is to reduce the amount of time and memory consumption, while still accomplishing the goal of improving the algorithm's performance. The combination of keccak and AES techniques that was suggested is applied to the data and the effectiveness of the task that has been suggested is evaluated with regard to the Avalanche effect, the amount of time it takes, and the memory overhead it creates. When the obtained results are compared with those of algorithms with similar functionality, the performance of the suggested method is shown to be superior. In the future, this work will be expanded, in order to make it operate with digital images on cloud storage.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] Rimal BP, Choi E, Lumb I. A taxonomy and survey of cloud computing systems. 2009 fifth international joint conference on INC, IMS and IDC; 2009, August. pp. 44–51. Ieee.

[2] Zhou, M., Zhang, R., Xie, W., et al. Security and privacy in cloud computing: a survey. In: 2010 sixth international conference on semantics, knowledge and grids. Beijing, China: IEEE; 2010, November. pp. 105–112. DOI:10.1109/SKG.2010.19

[3] Zhou L, Varadharajan V, Hitchens M. Trust enhanced cryptographic role-based access control for secure cloud data storage. IEEE Trans Inf Forensics Secur. 2015;10(11):2381–2395.

[4] Ali M, Malik SU, Khan SU. DaSCE: data security for cloud environment with semi-trusted third party. IEEE Trans Cloud Comp. 2015;5(4):642–655.

[5] Tian J, Jing X. A lightweight secure auditing scheme for shared data in cloud storage. IEEE Access. 2019;7:68071–68082. DOI:10.1109/ACCESS.2019.2916889

[6] Al Hamid HA, Rahman SMM, Hossain MS, et al. A security model for preserving the privacy of medical big data in a healthcare cloud using a fog computing facility with pairing-based cryptography. IEEE Access. 2017;5:22313–22328.

[7] Yu J, Lu P, Zhu Y, et al. Toward secure multikeyword top-k retrieval over encrypted cloud data. IEEE Trans Dependable Secure Comput. 2013;10(4):239–250.

[8] Wang Y, Wu Q, Qin B, et al. Identity-based data outsourcing with comprehensive auditing in clouds. IEEE Trans Inf Forensics Secur. 2016;12(4):940–952.

[9] Varadharajan V, Tupakula U. Security as a service model for cloud environment. IEEE Trans Netw Serv Manage. 2014;11(1):60–75.

[10] Wang H, He D, Tang S. Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. IEEE Trans Inf Forensics Secur. 2016;11(6):1165–1176.

[11] Zhu H, Yuan Y, Chen Y, et al. A secure and efficient data integrity verification scheme for cloud-IoT based on short signature. IEEE Access. 2019;7:90036–90044.

[12] Gonzales D, Kaplan JM, Saltzman E, et al. Cloud-trust—A security assessment model for infrastructure as a service (IaaS) clouds. IEEE Trans Cloud Comp. 2015;5(3):523–536.

[13] Shen J, Zhou T, He D, et al. Block design-based key agreement for group data sharing in cloud computing. IEEE Trans Depend Secure Comput. 2017;16(6):996–1010.

[14] Ni J, Zhang K, Yu Y, et al. Identity-based provable data possession from RSA assumption for secure cloud storage. IEEE Trans Depend Secure Comput. 2020;19(3):1753–1769.

[15] Li H, Liu L, Lan C, et al. Lattice-based privacy-preserving and forward-secure cloud storage public auditing scheme. IEEE Access. 2020;8:86797–86809.

[16] Zhang X, Zhao J, Xu C, et al. Dopiv: post-quantum secure identity-based data outsourcing with public integrity verification in cloud storage. IEEE Trans Serv Comp. 2019;15(1):334–345.

[17] Guo C, Su S, Choo KKR, et al. A provably secure and efficient range query scheme for outsourced encrypted uncertain data from cloud-based internet of things systems. IEEE Internet Things J. 2021;9(3):1848–1860.

[18] Xiong H, Choo KKR, Vasilakos AV. Revocable identity-based access control for big data with verifiable

outsourced computing. IEEE Trans Big Data. 2017;8(1): 1–13.
[19] Wang H, Zheng Z, Wu L, et al. New directly revocable attribute-based encryption scheme and its application in cloud storage environment. Cluster Comput. 2017;20:2385–2392.
[20] Sifah EB, Xia Q, Agyekum KOBO, et al. A blockchain approach to ensuring provenance to outsourced cloud data in a sharing ecosystem. IEEE Syst J. 2021;16(1): 1673–1684.
[21] Ra G, Kim D, Seo D, et al. A federated framework for fine-grained cloud access control for intelligent big data analytic by service providers. IEEE Access. 2021;9:47084–47095.
[22] Li Y, Yu Y, Chen R, et al. Integritychain: provable data possession for decentralized storage. IEEE J Sel Areas Commun. 2020;38(6):1205–1217.
[23] Khashan OA. Secure outsourcing and sharing of cloud data using a user-side encrypted file system. IEEE Access. 2020;8:210855–210867.
[24] Rao L, Zhang H, Tu T. Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated merkle hash tree. IEEE Trans Serv Comp. 2017;13(3):451–463.
[25] Liu L, Su J, Liu X, et al. Toward highly secure yet efficient KNN classification scheme on outsourced cloud data. IEEE Internet Things J. 2019;6(6):9841–9852.
[26] Chen Y, Mao Q, Wang B, et al. Privacy-Preserving multiclass support vector machine model on medical diagnosis. IEEE J Biomed Health Inform. 2022;26(7):3342–3353.
[27] Gupta I, Singh AK, Lee CN, et al. (2022). Secure data storage and sharing techniques for data protection in cloud environments: A systematic review, analysis, and future directions. IEEE Access.
[28] Mahajan HB, Rashid AS, Junnarkar AA, et al. Integration of healthcare 4.0 and blockchain into secure cloud-based electronic health records systems. Appl Nanosci. 2023;13:2329–2342. DOI:10.1007/s13204-021-02164-0
[29] Yang C, Song B, Ding Y, et al. Efficient data integrity auditing supporting provable data update for secure cloud storage. Wirel Commun Mob Comput. 2022:1–12. DOI:10.1155/2022/5721917
[30] Mohiyuddin A, Javed AR, Chakraborty C, et al. Secure cloud storage for medical IoT data using adaptive neuro-fuzzy inference system. Int J Fuzzy Syst. 2022;24(2):1203–1215.
[31] Seth B, Dalal S, Jaglan V, et al. Integrating encryption techniques for secure data storage in the cloud. Trans Emerg Telecommun Techn. 2022;33(4):1–24.
[32] Sajid, F., Hassan, M. A., Khan, A. A., et al. (2022). Secure and efficient data storage operations by using intelligent classification technique and RSA algorithm in IoT-based cloud computing. Sci Program; 2022:1–10. DOI:10.1155/2022/2195646
[33] Ullah Z, Raza B, Shah H, et al. Towards blockchain-based secure storage and trusted data sharing scheme for IoT environment. IEEE Access. 2022;10:36978–36994.