Received: February 6, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

TSS – Twin Layered Security Scheme for Cloud Storage to Preserve Data Integrity

M. B. Jayalekshmi[1]* **S. H. Krishnaveni[2]**

[1]*Noorul Islam University, Tamilnadu, India*
[2] *Baselios Mathews II College of Engineering, Kerala, India*
* Corresponding author's Email: mbjayalekshmi@gmail.com

Abstract: Cloud computing is a paradigm which offers a range of services to the cloud users. Cloud storage is the most popular service, as the data owners are freed up from the data management and storage overhead. However, the data owners are concerned about the security of the data. In order to address this issue, this article presents a twin layered security scheme, which guarantees the security of the data and thus the data integrity is preserved. The proposed work involves two building blocks, which are hashing and encryption. The input data is separated into several blocks, followed by which a tree is constructed. This is followed by the computation of hash codes, which is done by the keccak-256 hash function. The hash codes are then encrypted by Advanced Encryption Standard (AES) (256) with a dynamic key being generated by CodeIgniter. Thus, the security of the system is tightened and misuse of the data is strictly denied. The performance of the proposed work is tested with respect to throughput, memory and time consumption. The experimental results are reasonable, when compared to the analogous algorithms.

Keywords: Cloud security, Hashing, Encryption, Data integrity.

1. Introduction

All Today's world deals with enormous amount of data each and every second of time. Most of the data are claimed to be confidential, as they comprise personal, health and financial information. The amount of data escalates with respect to time and it is very difficult for the organizations to store and manage the data. Besides this, the organizations must line up with the storage equipments, which is high-priced.

Data management is another noteworthy annoyance for the organizations. The concept of cloud computing is a compliment to the mid and small scale organizations, as it is the best choice to avoid sparing a huge fund for data storage purposes. Additionally, cloud computing provides a bundle of attractive features such as on-demand services, easy and reasonable payment options, simpler data maintenance and so on.

These features kindle the data owners to outsource their own data to the cloud storage. By this way, the data owners enjoy the benefits of reduced overhead with respect to storage and data management [1-5]. However, the data owners feel reluctant to provide the confidential data to the unreliable cloud service provider, as the data owners cannot have the direct control of data [6-10]. This introduces several security related issues such as data confidentiality, integrity and availability. For instance, certain clouds have misbehaved by removing or modifying the rarely accessed data [11-14]. The same kind of issue has been reported against Amazon S3, Gmail, and Sidekick for failure, email removal and disaster respectively [15, 16]. Thus, it is necessary to safeguard the consistency of the data and the integrity of the data has to be maintained.

The main intention of this research article is to present a twin layered security mechanism, which preserves the data integrity of cloud storage. The proposed work verifies the data integrity of the cloud stored data by employing the keccak's hash function and Advanced Encryption Standard (AES)

for carrying out the encryption operation. Thus, the proposed work provides twin layered security. The performance of the proposed work is analysed against the SHA-3 member algorithms such as BLAKE, SKEIN and SHA2 with respect to throughput and memory. Besides this, the time consumption for encryption is also presented. The highlights of this article are listed as follows.

- Twin layered security mechanism is presented for the cloud storage, which involves hashing and encryption.
- The organization of data nodes in structured tree format allows updating any part of data at any instant of time.
- No table look up is required and no tables are maintained. This reduces the memory consumption and improves the performance.
- Keccak's hash function is employed for hashing, as the degree of security and the processing speed are considerably improved.
- The proposed work is free from any arithmetic and rotation operations.
- AES algorithm is utilized for encryption, as it is known for its speed and efficiency.
- On analysing the performance of the proposed work, it shows surprising results when compared to the analogous techniques.

In substance, this article tightens the security by introducing two layers of security, which are hashing and encryption techniques. The data are organized in the form of tree, which makes it easy to perform operations such as data update, insert and delete. The experimental results of the proposed work are compared with the member algorithms of SHA-3 with respect to throughput, memory consumption and execution time.

The rest of the article is systematized in the following way. Section 2 analyses the related literature with respect to data integrity in cloud storage. The proposed technique is elaborated in section 3. The efficiency of the proposed work is examined in the section 4. Lastly, the concluding remarks are drawn.

2. Review of literature

This section contemplates to review the related literature with respect to data security in cloud storage.

As the growth of cryptographic hash functioning techniques progresses, the techniques to break the hash functions are also progressing. SHA is a set of hash functions approved by National Institute of Standards and Technology (NIST). SHA-1 is the initially proposed algorithm, which is of 160 bits. The famous SHA-1 has been broken by Wang [17]. Thus, the NIST planned to check the current status of cryptographic hash algorithms. This step is followed by the introduction of a new family of hashing algorithms, which is SHA-2.

SHA-2 algorithms can utilize four different block sizes such as 224, 256, 384 and 512. SHA-2 algorithms are more secure than the SHA-1 group of algorithms and are appropriate for commercial purposes. The major drawback being cited by the SHA-2 algorithms is the interoperability [18].

The SHA-2 algorithms started to face several attacks and thus, the NIST again circulates a proposal for finding new secure and compatible algorithms. Several algorithms were proposed as members of SHA-3 and they are Blake, Skein, Keccak, JH and Grostl. Blake algorithm entered the final round of SHA-3 algorithm selection and is a family of four different hash functions such as Blake-224, Blake-256, Blake-384 and Blake-512. Blake utilizes minimal resources and is faster both in terms of software and hardware [17,19]. However, the security of this algorithm is not up to the mark.

The Groestl algorithm is based on bytes and is completely different from the general design principle of SHA group of algorithms [18]. Some of the noteworthy advantages of Groestl algorithm are its security, simplicity, side channel resistivity and parallelism. However, this algorithm is susceptible to collision attacks. The detailed description of the Groestl algorithm can be found in [19].

JH is another important member of SHA-3 algorithms and it consists of four different hash functions. They are JH-224,256,384 and 512. The major operations involved in JH are permutation, substitution and XOR. The design of JH is elaborated in [19] and the performance of JH is poorer than keccak. Skein which is a constituent of SHA-3 family algorithms comes in three different sizes, which are 256, 512 and 1024 bits. The building blocks of Skein algorithm are Threefish, Unique Block Iteration (UBI) and Optional Argument System [20,21].

Keccak algorithm is the most efficient algorithm in the SHA-3 group of hash algorithms. This algorithm relies on the process of constructing sponge. Keccak is proved to be secure against generic attacks and some of the key merits of keccak are simplicity and flexibility. The keccak algorithm is well explained in [19].

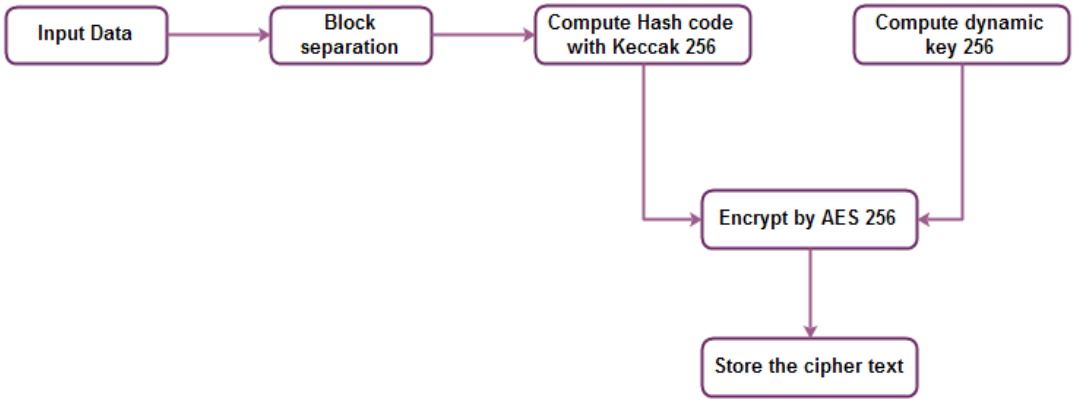

Figure.1 Overview of twin layered security scheme

Motivated by the above works, this article aims to present a twin layered security scheme for cloud storage. The original keccak algorithm is modified to some extent and is clubbed with the AES encryption. This results in increased security and resistivity against security attacks. The proposed security model is elaborated in the forthcoming section.

3. Proposed twin layered security scheme

This part of the article presents the proposed twin layered security mechanism. The major entities being participated in this security model are data owners (DO), cloud service providers (CSP) and integrity checkers (IC). Data owner is an important entity, who outsource or handover the confidential data to the cloud server. Cloud service providers are commercial entities, which provide a range of services to the cloud users. In this case, the focus is on storage service.

In order to get rid of management and storage complications, the data owners outsource the data to the commercial cloud service provider. These two parties share a Service Level Agreement (SLA), which clarifies the terms of the service being provided. Though, the SLAs are violated certain times and this leads to several troubles. This issue is handled by the entity named integrity checker, which checks the consistency of the outsourced and the cloud stored data. The overall flow of the twin layered security scheme is depicted in Fig.1.

For achieving this, the entire work is compartmentalized into

- Hashing phase
- Encryption phase

These phases are again broken down into several sub-phases. The keccak's hash function is applied on the to-be outsourced data, in order to obtain the hash code. This is followed by the generation of dynamic key, which is an important weapon for the data owners. The dynamic key is not shared with any other parties. The AES encryption algorithm takes the hash code and dynamic key as the input and releases the encrypted version. Thus, the process of decryption is possible only when the dynamic key is available. This ensures the security of the entire system and the forthcoming sections elaborate the system.

3.1 Hashing phase

The major objective of the process of hashing is to enhance the security of the model. Keccak's hash function is utilized for achieving this task. The two important stages of this phase are block separation and hash value computation. Keccak is a group of hash functions and the heart of keccak is the sponge establishment.

The sponge function is attained by three important building blocks, which are bit rate (br), capacity (c) and diversifier (d). The bit rate and capacity together decides the degree of the permutation. The main usage of diversifier is to differentiate between the entities of keccak.

The keccak's function accepts the entire data to be outsourced and breaks it into multiple blocks by employing the sponge function. Thus, the input can be of any size and the outcome of the sponge function is a set of blocks, whose size is fixed by the user. The bit rate and capacity collectively determines the security and the speed of the system. The bit rate is directly proportional to the security of the system.

On the other hand, the speed of the system is improved by increasing the capacity value. This proposed security model works with two different output lengths 256 and 512, in order to check the execution speed of the system.

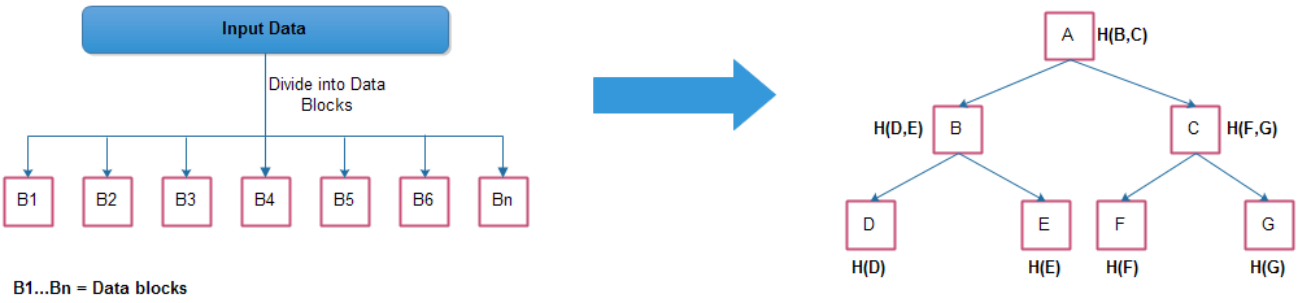

Figure.2 Hashing phase

Keccak enforces a policy that the summation of bit rate and capacity must be 1600. This function performs all the operations with respect to bytes, so as to overcome the bit attacks. The idea behind the hashing phase is presented in Fig.2.

As this function operates over bytes, an issue may arise when the input data cannot be exactly divided into bytes. In this case, certain number of bits is present and these bits are handled separately. The remaining count of bits is called as trailing bits, which are appended at the last node of the tree. Let D be the data which is to be outsourced to the cloud. The proposed approach divides the data D into several data blocks, which can be denoted as

$$D = \{DB_1, DB_2, DB_3, \ldots DB_n\} \quad (1)$$

All the data blocks are organized in the form of standard binary tree. The hash codes of the child nodes are computed and are appended to the parent node. This process continues till the root node is reached.

$$H(D) = \{H(DB_1), H(DB_2), H(DB_3), \ldots H(DB_n)\} \quad (2)$$

When the input data is separated into data blocks, a tree is formed with several nodes. The tree can grow with respect to the size of the data and no restriction is enforced regarding the height of the tree. Thus, the height of the tree is directly proportional to the size of the input data. The entire process is presented in the following part.

Hashing phase
//Block separation
Input : data, ol
Output : Data blocks
Begin
Decompose data w.r.t ol;
If ((size(data)%8)==0) then
Construct tree;
Goto Hash_Computation;
Else
Collect trailing bits in a separate block;
End;
//Hash Computation
Input: Data blocks
Output: Hash code
Begin
Perform hash operation over leaf nodes;
Do
Concatenate hash codes of nodes with same parent;
until the root node is reached;
End;

In the above algorithm, *ol* is the output length and trailing bits are the remaining bits after the separation of input data into data blocks. Initially, the hash value is computed for all the leaf nodes. The hash values of the leaf nodes are concatenated, if the leaf nodes are the children of the same parent. This process continues till the root node is reached. Thus, the hash code is computed and is stored in the root node.

3.2 Encryption phase

After the construction of tree with hash codes, the security of the system is even more tightened with the concept of encryption. In this phase, the famous encryption algorithm AES is utilized. AES is known for its execution speed and high degree of security. In order to enhance the security, the hash codes are encrypted by the dynamic key. The dynamic key is never shared by the data owner to any party. In this work, the dynamic key is generated by the CodeIgniter key generator, which is of size 256 bits.

This work prefers AES for encryption, as it operates over bytes rather than bits. In this work, 256 bit key is employed for AES, which results in 14 rounds of encryption. As the number of rounds increases, the security of the system is also improved. Each and every round exploits a 128 bit key, which is computed by the AES key itself. The algorithm for encryption phase is presented below.

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;

Each round of AES involves several significant operations such as substituting bytes, shift rows, mix columns and add round key. Initially, the input bytes are loaded into the S box, which is in the form of a matrix. The rows are then shifted with respect to positional constraints. The columns are then shuffled by replacing the original bytes into different bytes and this operation does not take place at the 14th round alone. The final components of the matrix are performed XOR operation with the round key. This process is repeated for 14 rounds, in order to provide the cipher text.

As the dynamic key is never shared, the security of the system is far more improved. Besides this, the plain text is not directly encrypted but the computed hash codes are encrypted. Thus, this work provides a twin layered security.

When the data owner needs to perform integrity check, the AES algorithm is decrypted with the secret dynamic key, such that the hash code is obtained. The hash codes can then be compared, in order to check the data integrity. The integrity check can be performed by the data owner at any instant of time by challenging the cloud server. The next section aims to present a sample work principle of the proposed work.

3.2 Sample work principle of twin layered security scheme

To exemplify this concept, a small portion of data from the huge amount of data is considered. This makes sense that this small portion of data contributes the role of a node in a big tree of data. The sample work principle is just added for understanding purpose.

The small part of data being considered is as follows. The sample data is of about 2386 bytes. In order to explain the concept, the data is divided into several 500 bytes. Thus, the tree constitutes 6 nodes and the last node is loaded with the remaining 386 bytes. The structure of the data is represented as tree in Fig.3.

Initially, the hash code of 6th node is saved in node 3. The hash codes of 4th and 5th nodes are computed and stored in node 2. Similarly, the hash codes of 2nd and 3rd nodes are saved in node 1, which is the root node. Thus, a complete tree is considered. The count of hash codes is reduced, as the hash code is computed only for the children nodes, which share the same parent.

The root node stores only the complete hash code and it doesn't possess any data. The hash code is computed by Keccaks hash function of size 256 bits. This is followed by the generation of a dynamic key with CodeIgniter and the hash code is encrypted with the so computed key, by means of AES. The 256 bit hash codes of the nodes are presented below.

Computed hash codes by Keccak (256 bit) [HC]
Node 6:
bf807b6a14256c3e49c479060f83d24a941e4e9f3a1d
aadbc052e7e87c7079b7

Nodes 4 and 5:
0f1b466a489726a7763cfb48c2443206556320ed174
912a6429b08445058455a

Nodes 2 and 3:
bebd623270d374cf11e7dab1ca385822a793dcb2c47
4ec0e98d0dd2ee5cf23b1
Finally, the hash code of nodes 2 and 3 is stored in the root node. After this process, a dynamic key (256 bit) is generated and is presented below.

Generated dynamic key [DK]:
agC7Z5TKA6mi6trYCr7NeB4LE6mxqhug

AES encryption (256 bit) [HC+DK] :
Node 6:
D/jDd8VKWvVByEcSaOrowYiEJMZpV5bJBp29I
XYVWMqoySCxnwOGuEZ6zvSLiZttpvefR7Qvss4
kkladqyKAUA==

Nodes 4 and 5:
hqq4Na122rFXfEB2+UE2sCvYexrcs0MosAedjrES
VxPn6u/Re8mcTcG4FNO/Vn9hMc6B5FMtMaZtIi
+y+dPI7A==

Nodes 2 and 3:
6gh7SOQc1wqn7iNmmrwfubvZmN7A9R95LAf/1P
k/BAt1SBJDmDWS8sr/G1Gfx5q5c7ubboiqEsuY7Z
ZLurDFPg==

The hash codes of the nodes can be obtained only when the dynamic key is known. Even when the dynamic key is known, it is difficult to break the keccak's security. So far, no security breaches are reported for Keccak. Besides this, we employ AES

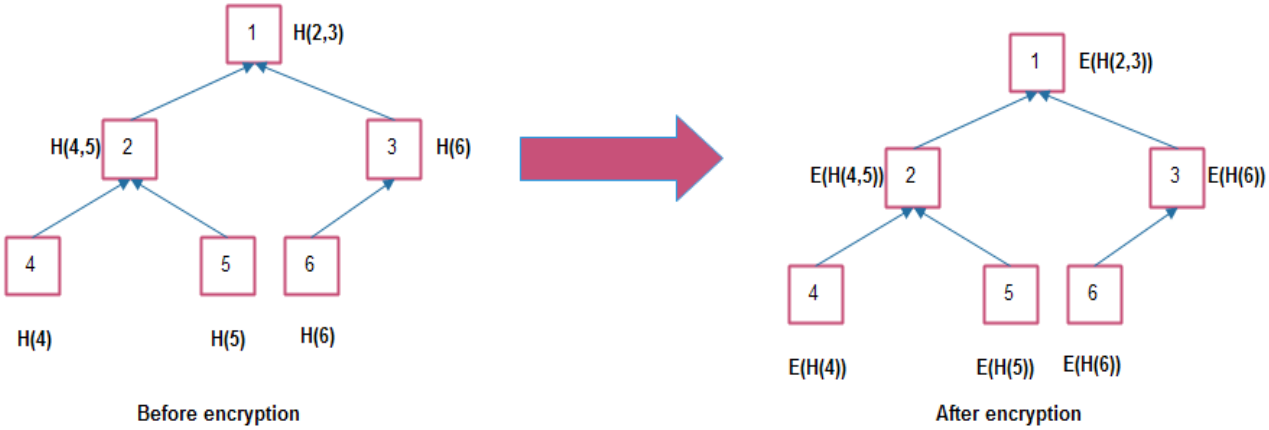

Figure.3 Structure of sample data

algorithm of 256 bits, which takes 14 rounds to arrive at the cipher text.

Thus, the security of the system is believed to the safest at the expense of complex computation. By this way, the integrity of the
data is preserved. Most of existing systems rely on the computation of hash codes alone. However, this work adds another line of security which is impossible to break.

4. Performance analysis

The performance of the twin layered security model is analysed by comparing with the SHA 3 member algorithms such as BLAKE [17], SKEIN [20] and SHA2 [19] in terms of throughput and memory consumption. Throughput is an important performance metric for determining the speed of the algorithm, which makes sense that the capability of the algorithm to deal with the data rate. It is measured in terms of Mbps (mega bits per second). The memory requirement is also taken as a measure to prove the effectiveness of the proposed security model. Finally, the time consumption for encryption is taken into account and the experimental results are presented below.

The throughput of any security model must be greater, such that the execution time of the security model can be minimized. In case of minimal throughput, the process of execution would take more time, which consumes more resources. The proposed model is tested for the throughput for many different data and it shows satisfactory results. The following graphical results show the throughput rate being shown for three different data with varied sizes. Among all the comparative algorithms SHA 2 performs very poor in all the cases. Keccak performs well in all the cases and is followed by Skein. The throughput of keccak is greater because it processes block by block.

This is followed by analysing the memory consumption of the proposed twin layered security scheme. The memory consumption is tested against different volumes of data with the same comparative algorithms. The memory consumption of the proposed model is observed to be minimal. The minimal memory consumption is because of the usage of blocks and the storage of hash codes in place of the original data. Besides this, the hash codes are not needed to be computed for individual nodes, but the hash codes are computed for the children nodes of the same parent. All these features reduce the memory consumption of the proposed work.

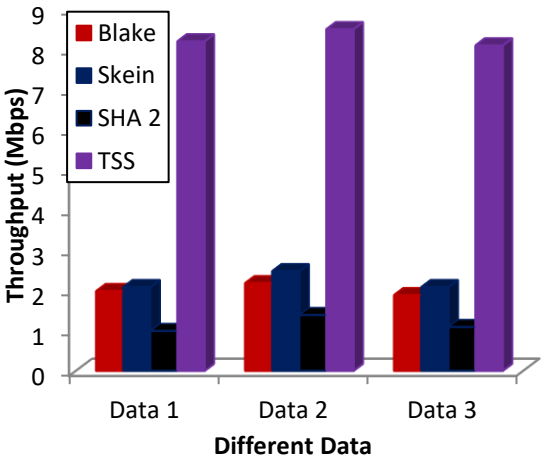

Figure.4 Throughput analysis

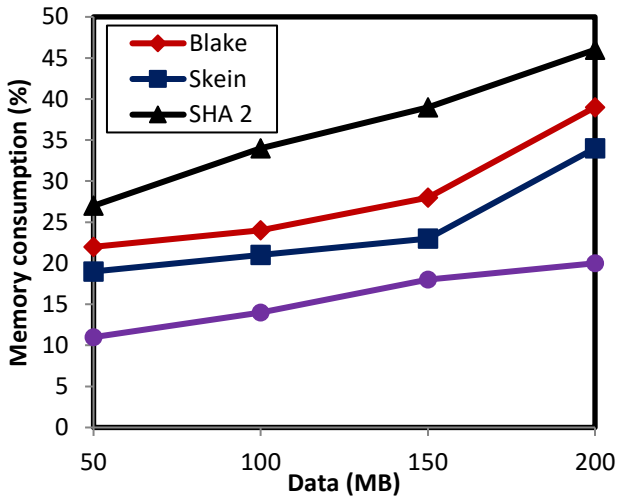

Figure.5 Memory consumption analysis

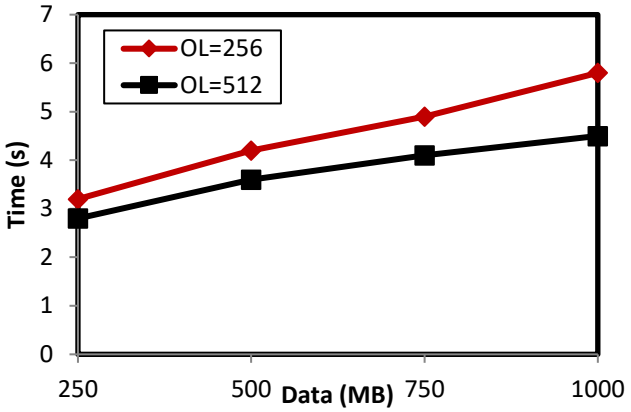

Figure.6 Execution time analysis

The execution time is another important performance metric of any algorithm. The execution time must be as minimal as possible, such that the latency is minimized. It is not feasible for a well-performing algorithm to have extended execution time. The execution time of the proposed twin layered security scheme is tested by differentiating the data size. This experiment is carried out in two different scenarios by varying the output length. The output length is varied as 256 and 512 bytes.

From the above graph, it can be observed that the length of output acts a deciding factor of the execution time. The above result is attained with the Intel core i7 processor. When the output length is set as 256 bytes, the process of execution consumes more time than when the output length is fixed as 512 bytes. The output length can be increased when speed is given more importance than the security. When the output length is set as 256, the br, c and d are set as 1088, 512 and 32 respectively. Alternately, when the output length is fixed as 512, the values of br, c and d are set as 576, 1024 and 64 respectively. Thus, when the count of rounds increases, the security of the system is improved.

Thus, the main objective of the paper to present a twin layered security model for achieving data integrity in cloud storage is achieved. The proposed security model is examined with several performance metrics and the proposed work outperforms the comparative algorithms with greater throughput, minimal memory and time consumption.

5. Conclusion

This article presents a twin layered security model for the cloud storage, in order to preserve the data against numerous security threats. The data integrity is preserved as the security of the system is tightened. The entire model is divided into two important phases, which are hashing and encryption phase. As both hashing and encryption are present in the model, the security is guaranteed at the expense of computation.

Both these phases are achieved by keccak's hash function and AES algorithm. Finally, the performance of the twin layered security scheme is analysed and the experimental results show the efficacy of the work. The performance of the proposed work is analysed in terms of throughput, memory consumption and execution time. Though the proposed approach is found to be secure and memory conserving, the computational complexity is a bit high. In future, we plan to incorporate the concept of multi-clouds with data replicas.

References

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin and I.Stoica, "Above the Clouds: A View of Cloud Computing", *Communications of the ACM*, Vol. 53, No. 4, pp. 50-58, 2010.

[2] C. Kai, H. Chengchen, Z. Xin, Z. Kai, C. Yan, and A.V. Vasilakos, "Survey on routing in data centers: insights and future directions", *IEEE Network*, Vol.25, No. 4, pp. 6-10, 2011.

[3] W. Lin, Z. Fa, Z. Kai, A.V. Vasilakos, R. Shaolei, and L. Zhiyong, "Energy-Efficient Flow Scheduling and Routing with Hard Deadlines in Data Center Networks", In*: Proc. of IEEE International Conference on Distributed Computing Systems (ICDCS*), Madrid, pp. 248-257, 2014.

[4] M.R. Rahimi, N. Venkatasubramanian, and A.V. Vasilakos, "MuSIC: Mobility-Aware Optimal Service Allocation in Mobile Cloud Computing", In: *IEEE Sixth International Conference on*

Cloud Computing (CLOUD), Santa Clara, CA, pp. 75-82, 2013.

[5] G. Wei, A. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services", *The Journal of Supercomputing*, Vol. 54, No. 2, pp. 252-269, 2010.

[6] W. Cong, R. Kui, L. Wenjing and L. Jin, "Toward publicly auditable secure cloud data storage services", *IEEE Network*, Vol. 24, No. 4, pp. 19-24, 2010.

[7] A. Khan, M.L.M. Kiah, S. Madani, M. Ali, A. Khan and S. Shamshirband, "Incremental proxy re-encryption scheme for mobile cloud computing environment", *The Journal of Supercomputing*, Vol. 68, No. 2, pp. 624-651, 2014.

[8] W. Lin, Z. Fa, J. Arjona Aroca, A.V. Vasilakos, Z. Kai, H. Chenying, L. Dan and L. Zhiyong, "GreenDCN: A General Framework for Achieving Energy Efficiency in Data Center Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 32, No. 1, pp. 4-15, 2014.

[9] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen and A.V. Vasilakos, "Security and privacy for storage and computation in cloud computing", *Information Sciences*, Vol. 258, pp. 371-386, 2014.

[10] Z. Xin, Z. Fanfu, Z. Xinyu, S. Haiyang, A. Perrig, A.V. Vasilakos and H. Guan, "DFL: Secure and Practical Fault Localization for Datacenter Networks", *IEEE/ACM Transactions on Networking*, Vol. 22, No. 4, pp. 1218-1231, 2014.

[11] M. Ali, S.U. Khan and A.V. Vasilakos, "Security in cloud computing: Opportunities and challenges", *Information Sciences*, Vol. 305, pp. 357-383, 2015.

[12] S. Shamshirband, N.B. Anuar, M.L.M. Kiah, and A. Patel, "An appraisal and design of a multi-agent system based cooperative wireless intrusion detection computational intelligence technique", *Engineering Applications of Artificial Intelligence*, Vol. 26, No. 9, pp. 2105-2127, 2013.

[13] C. Wang, K. Ren, W. Lou and J. Li, "Toward publicly auditable secure cloud data storage services", *IEEE Network*, Vol. 24, No. 4, 19-24, 2010.

[14] K. Yang and X. Jia, "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing", *IEEE Transactions on Paralel and Distributed Systems*, Vol. 24, No. 9, pp. 1717 – 1726, 2013.

[15] T.Armerding, "The 15 worst data security breaches of the 21st century", In *: COS Security and Risk*, Csoonline, 2012.

[16] G.M. Stevens, "Data Security Breach Notification Laws", In: *Congressional Research Service*, 2012.

[17] J.P. Aumasson, L. Henzen, W. Meier and R. C.-W. Phan, "SHA-3 Proposal BLAKE," *NIST (Round 3)*, University of California Santa Barbara, Santa Barbara, 2010.

[18] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, C. Rechberger, M. Schläffer and S. S. Thomsen, "Grøstl—A SHA-3 Candidate," In : *NIST*, University of California Santa Barbara, Santa Barbara, 2011.

[19] X. Guo, M. Srivastav, S. Huang, D. Ganta, M. B. Henry, L. Nazhandali and P. Schaumont, "Silicon Implementation of SHA-3 Finalists: BLAKE, Grøstl, JH, Keccak and *Skein," Center for Embedded Systems for Critical Applications (CESCA) Bradley Department of Electrical and Computer Engineering Virginia Tech*, Blacksburg, 2010.

[20] X. Guo, M. Srivistav, S. Huang, D. Ganta, M. Henry, L. Nazhandali and P. Schaumont, "Silicon Implementation of SHA-3 Finalists: BLAKE, Grøstl, JH, Keccak and Skein," *ECRYPT II Hash Workshop 2011*, Tallinn, May 2011.

[21] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas and J. Walker, "The Skein Hash Function Family," *NIST Cryptographic Hash Algorithm Competition*, University of California Santa Barbara, Santa Barbara, 2008.