Scientific Research Publishing

Circuits and Systems, 2016, 7, 3626-3644
http://www.scirp.org/journal/cs
ISSN Online: 2153-1293
ISSN Print: 2153-1285

Block Level Data Integrity Assurance Using Matrix Dialing Method towards High Performance Data Security on Cloud Storage

P. Premkumar[1], D. Shanthi[2]

[1]Department of Computer Science and Engineering, K.L.N. College of Engineering, Pottapalayam, India
[2]Department of Computer Science and Engineering, PSNA College of Engineering & Technology, Dindigul, India
Email: Premkumar20041971@gmail.com, dshan71@gmail.com

How to cite this paper: Premkumar, P. and Shanthi, D. (2016) Block Level Data Integrity Assurance Using Matrix Dialing Method towards High Performance Data Security on Cloud Storage. *Circuits and Systems*, **7**, 3626-3644.
http://dx.doi.org/10.4236/cs.2016.711307

Received: May 10, 2016
Accepted: May 25, 2016
Published: September 13, 2016

Abstract

Data outsourcing through cloud storage enables the users to share on-demand resources with cost effective IT services but several security issues arise like confidentiality, integrity and authentication. Each of them plays an important role in the successful achievement of the other. In cloud computing data integrity assurance is one of the major challenges because the user has no control over the security mechanism to protect the data. Data integrity insures that data received are the same as data stored. It is a result of data security but data integrity refers to validity and accuracy of data rather than protect the data. Data security refers to protection of data against unauthorized access, modification or corruption and it is necessary to ensure data integrity. This paper proposed a new approach using Matrix Dialing Method in block level to enhance the performance of both data integrity and data security without using Third Party Auditor (TPA). In this approach, the data are partitioned into number of blocks and each block converted into a square matrix. Determinant factor of each matrix is generated dynamically to ensure data integrity. This model also implements a combination of AES algorithm and SHA-1 algorithm for digital signature generation. Data coloring on digital signature is applied to ensure data security with better performance. The performance analysis using cloud simulator shows that the proposed scheme is highly efficient and secure as it overcomes the limitations of previous approaches of data security using encryption and decryption algorithms and data integrity assurance using TPA due to server computation time and accuracy.

Keywords

Cloud Computing, Data Integrity, Data Security, SHA-1, Digital Signature, AES, Encryption and Decryption

1. Introduction

Cloud computing is a modern computing paradigm in which scalable resources are shared dynamically as various services over the internet [1]. Cloud storage services enable the user to enjoy with high capacity and quality storage with less overhead but it has many potential threats like data integrity, data availability, data privacy and so on. The two issues mainly occur while outsourcing the data using cloud storage is data integrity and data security due to unfaithful cloud service provider [2]. Data integrity is the form of protection of data against loss and damage caused by hardware, software and network failure [3] [4]. Normally data inaccuracy can occur either accidently through programming errors or maliciously through breaches or hacks. It is one of the important aspects among the other cloud storage issues because data integrity ensured that data are of quality, correctness, consistency, accuracy, security, confidentiality, reliability, and accessibility but assurance of data integrity in the cloud is a major challenge that is faced by today's cloud users [5]. It refers to assurance by the user that the data are not modified or corrupted by the service provider or other users. The performance of data integrity is measured by using the parameters like computation time, encryption time and decryption time, memory utilization and output size. While outsourcing their data using cloud storage does not maintain a local copy. Hence, cryptographic measures cannot be used directly to monitor the integrity of data and also downloading the data for monitoring integrity is not a viable solution. Therefore, an external Third Party Auditor (TPA) is required [6]. The TPA is an independent authority that has capabilities to monitor the integrity of outsourced data by the client and also inform on data corruption or loss, if any. But it requires separate memory and also takes more time for verification of data to ensure integrity of data; hence the overall performance is degraded. Nowadays, software professionals employ number of practices to ensure data integrity which includes data encryption, data backup, access controls, input validation, data checking, error detection and correction while transmitting and storing the data. The performance of data violation checking methods is affected due to communication overhead, memory overhead, key size, encryption time, decryption time, and computation time. The scope of the data integrity assurance mechanism can be classified into two levels: first is to prevent data corruption and second is to detect and correct data violation. This paper only focuses on detection of data violation. The algorithms and methods to ensure data integrity are discussed in [7]. In paper [8], certain degree of integrity assurance is provided by RAID technique but it operates only on binary data, takes more computation time and also the value of determinant factor is three bits long and hence needs large memory for storage. In paper [9], to evaluate the performance of the encryption algorithm for text files, it uses AES, DES [10] [11] and RSA algorithm and the parameters such as computation time, memory usage, and output bytes are considered. The time taken to convert the plain text into cipher text is known as encryption time. The decryption time is the time that a decryption algorithm takes to reproduce a plaintext from a cipher text. Comparing these three algorithms, RSA takes more time for computation [12]. The memory usage of each algorithm is

byte level. RSA requires more memory than AES and DES. In paper [13], various algorithms such as AES, 3DES, Blowfish and DES are discussed. Throughput is equal to total encrypted plaintext in bytes divided by the encryption time. Higher the throughput, higher will be the performance [14]. Asymmetric encryption techniques are slower than symmetric techniques, because they require more computational processing power. Also, Blowfish algorithm gives better performance than all other algorithms in terms of throughput [15] [16]. In paper [17] and paper [18], the performance evaluation of AES and Blowfish algorithms is discussed [19]. The parameters such as time consumption of packet size for 64 bit encodings and hexadecimal encodings, performance for encryption of text files and the throughput are considered. The result shows that Blowfish has better performance than AES in almost all the test cases. The paper is organized as follows: Section 1 describes introduction and related work. The proposed methodology is discussed in Section 2. Section 3 describes comparison of results and analysis. Section 4 concludes the paper.

2. Proposed Methodology

The proposed technique is based on the Determinant Factor (DF) approach to enhance both data integrity and security which involves the following steps:

Before transmitting the series of data, it is divided into N-matrices, where N is given by:

$$N = \frac{\text{total number of data}}{(d * d)}$$

where $(d \times d)$ is the number of elements per matrix. The determinant factor of each matrix is computed and appended with the data. At retrieving stage, it is compared with the determinant factor of the sender's data for data integrity assurance. But it is observed that there is one defect with this method. The DF is zero if any one of the rows is proportional to another row; the same is true for columns. Also, the DF does not change if some of the rows or some of the columns are interchanged. In addition, the DF is zero if any single row or column has zero values only. In order to alleviate this problem, a new technique is performed as given below: For each element of the matrix is reconstructed using matrix dialing method to formalize the original data matrix into a new matrix [20]. The determinants of both original and Dialing rotational matrices computed and appended with each matrix. For example, DF value for the following matrix is zero. After applying this new technique, DF value of the resultant matrix is not zero.

$$\begin{pmatrix} e1 & e2 & e3 \\ e4 & e5 & e6 \\ e7 & e8 & e9 \end{pmatrix} = \begin{pmatrix} e4 & e1 & e2 \\ e7 & e5 & e3 \\ e8 & e9 & e6 \end{pmatrix}$$

Then encrypted digital signature for each determinant factor is generated using the

combination of SHA-1 and AES algorithms. Finally, data coloring is applied on each digital signature before transmission or storing the data on cloud to enhance data security. At the receiver side, both determinants are recomputed again and also degenerate the Message Digest then compared with the sender's values. If there is a match, it ensures that there is no modification in the given data during the transmission otherwise particular block of data is to be violated. The results of the proposed system shows that block based matrix dialing method outperforms than other data integrity checking methods and also provides data privacy for securing the data from unauthorized users. Figure 1 shows the architecture of the proposed system.

Steps involved in Block based determinant approach is given below:

Sender's End

1) Data is taken as a string format. Each string is converted as bytes and the number of bytes that constitute a block is decided. Next bytes will be added and divided into number of blocks.

2) Convert each block of data into square matrix.

3) Find Determinant Factor (DF) for each matrix.

4) Construct a new matrix using Block Based Matrix Dialing Rotational method to ensure DF is not Zero.

5) Find DF for the matrix constructed in Step 4.

6) Generate Hash value is known as Message Digest using SHA-1for each DF calculated in Step 5.

7) Encrypt this Hash value using AES algorithm to generate Digital Signature.

8) Apply data coloring on each digital signature generated in the Step 7.

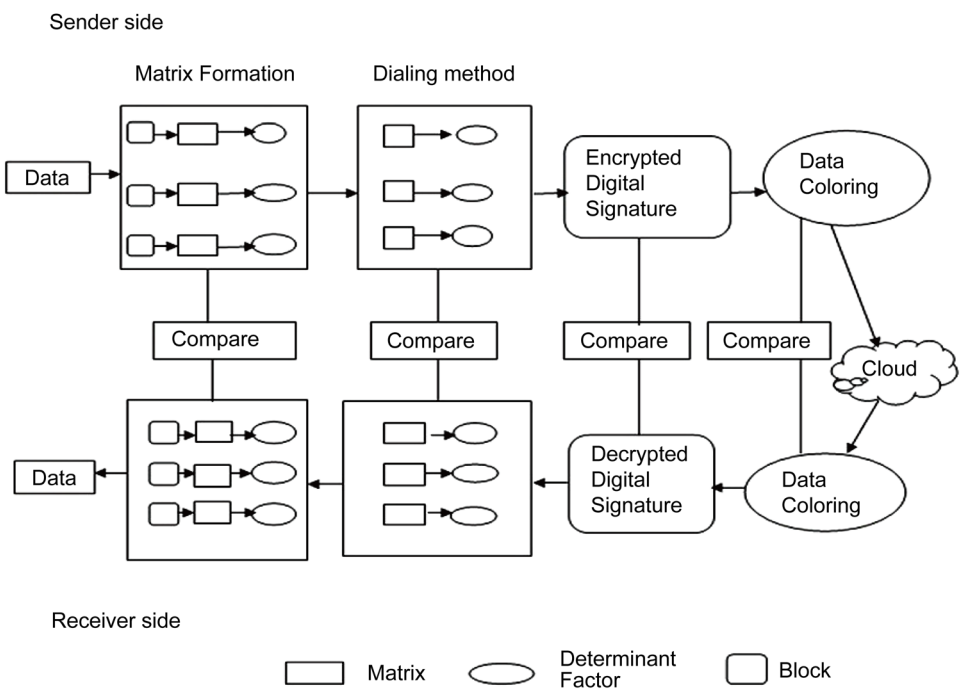

Figure 1. Architecture of the proposed system.

9) Store the colored data into cloud storage.

Receiver's End

1) Regenerate the colors from the colored data.

2) Decrypt the Message Digest.

3) Reconstruct the new matrix.

4) Calculate DF for the matrix constructed in Step 3.

5) Reconstruct the new matrix and calculate DF.

6) Compare the results obtained in steps 1, 2, 4, 5 respectively of Receiver's End with 8, 6, 5 and 3 of Sender's End.

7) If the results are same in all the steps mentioned in Step 6, then this ensures data integrity otherwise integrity of data is not attained *i.e.*, a particular block of data has been violated *i.e.* modified the given data by unauthorized users.

Steps 6, 7 of sender side and also Step 2 of receiver side is explained in detailed as given below:

Signed and Encryption

1) Sender sends a message as DF

2) Calculate Digest

$$\text{Digest} = [\text{Message}]_{\text{hash}}.$$

3) Sign the Digest

$$\text{Message} + [\text{Digest}]k_{\text{pri}} + k_{\text{pub}}.$$

4) Encrypt with Symmetric key

$$[\text{Message} + [\text{Digest}]k_{\text{pri}} + k_{\text{pub}} + k_{\text{sym}}.$$

5) Send signed and encrypted message to Recipient.

Here Steps 1), 2) and 3) are for Signature generation and Step 4) for encryption (AES algorithm).

Decrypt and Verifying message

1) $[\text{Message} + [\text{Digest}]k_{\text{pri}} + k_{\text{pub}]} + [k_{\text{sym}}]$.

2) Decrypt K_{sym} with receivers private key $[\text{Message} + [\text{Digest}]k_{\text{pri}} + k_{\text{pub}]} + k_{\text{sym}}$.

3) Decrypt Digest using Public key and also evaluate the Digest

$$\text{Digest} = [\text{Message}]_{\text{Hash}}.$$

4) Compare these two Digests.

If two digests viz., actual and expected digests are equal then the signature is verified. Here Steps 1, 2 and 3 are for Decryption and Step 4 for Verification.

The following steps are involved to generate encrypted digital signature; it described by Figure 2.

Step 1. The document will be crunched into fixed few lines by using SHA-1 algorithm to generate Message digest.

Step 2. At Sender side encrypt the message digest using its public key to generate digital signature.

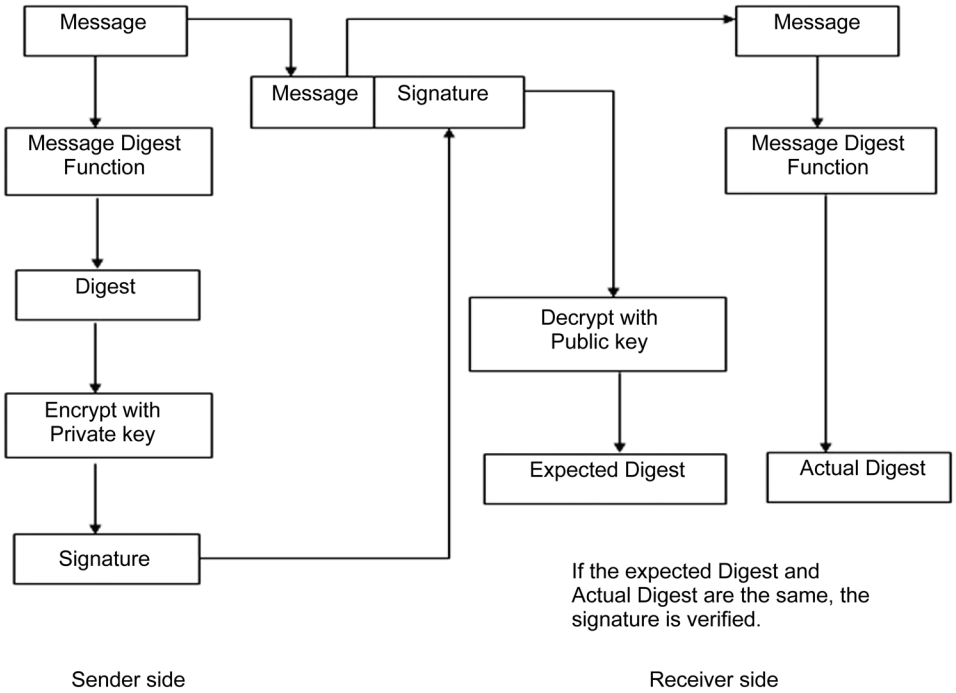

Figure 2. Block diagram for generation of digital signature.

Step 3. At Receiver side decrypt the message using their own private key.

Step 4. Regenerate the Message Digest.

Step 5. Finally the Signature is verified using Sender's public key.

Message digest function also called as hash function used to generate digital signature of the data which is known as message digest. SHA-1 algorithm is used to implement integrity of the message which produce message digest of size 128 bits. These are mathematical functions that process information to produce different message digest for each unique message. It processes the message and generates 128 bits message digest. The AES algorithm consists of the following steps and also it described by **Figure 3**.

Step 1: Add Padding to the end of the genuine message length is 64 bits and multiple of 512.

Step 2: Appending length. In this step the excluding length is calculated.

Step 3: Divide the input into 512-bit blocks. In this step the input is divided into 512 bit blocks.

Step 4: Initialize chaining variables. In this step chaining variables are initialized. In the proposed method 5 chaining variables are initialized each of size 32 bits giving a total of 160 bits.

Step 5: Process Blocksie., Copy the chaining variables, Divide the 512 into 16 sub blocks, Process 4 rounds of 20 steps each.

Step 6: Output Generation.

Further this algorithm is divided into 5 steps: Key Generation, Digital Signing, Encryption, Decryption and Signature Verification are discussed as below:

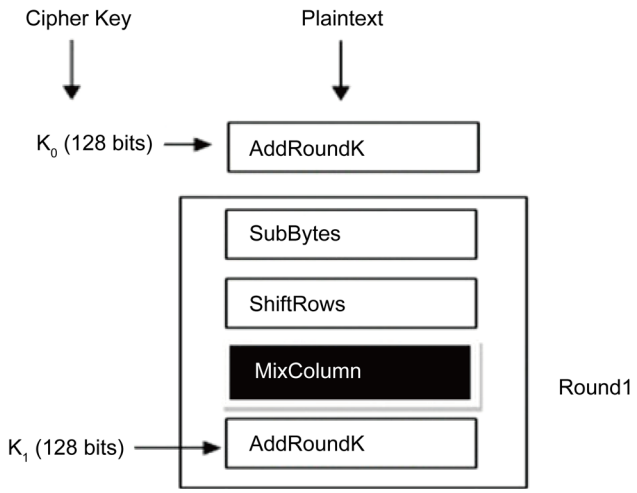

Figure 3. Block diagram for AES algorithm process.

Step 1: Key Generation

Different combinations of key size such as 128, 192 or 256 bits are used. To perform the AES algorithm, round keys must be generated from the user provided key. The Key Schedule of this algorithm provides 33 128-bit keys to be mixed with the text blocks during the Round function of the algorithm. First create 8 32-bit pre keys using the key provided by the user. The user's key is split every 32 bits to do this and then generate 132 intermediate keys using the following reoccurrence: for i from 0 to 131. The 33 round keys are generated from these intermediate keys by running through the S-Boxes and combining them into 128-bit blocks.

Step 2: Digital Signing

Generate message digest of the document to be send by using SHA-1 algorithm.

The digest is represented as an integer m.

Digital signature S is generated using the private key (n, d).

S = md mod n.

Sender sends this signature S to the recipient.

Step 3: Encryption

Sender represents the plain text message as a positive integer m.

It converts the message into encrypted form using the receiver's public key (e, n).

C = me mod n

Sender sends this encrypted message to the recipient. Here, n is the modulus and e is the encryption exponent.

Step 4: Decryption

Recipient does the following operation:

Using his private key (n, d), it converts the cipher text to plain text "m".

M = Cd mod n

where *d* is the secret exponent or decryption exponent.

Step 5: Signature Verification

Receiver does the followings to verify the signature:

An integer V is generated using the sender's public key (n, e) and signature S.

V = Se mod n

It extracts the message digest M1, from the integer V using the same SHA-1 algorithm. It computes the message digest M2 from the signature S. If both the message digests are identical *i.e.* M1 = M2, then signature is valid.

The block diagram for generating color coding is shown in Figure 4.

Each user is specified by a color that helps to protect and also avoids the manipulation of original data. Figure 4 shows the details involved in the data coloring process, which aims to associate a colored data with its own, whose user identification is also colored with the same expected value (*Ex*), *entropy* (*En*) and *hyperentropy* (*He*) identification characteristics. The cloud drops are added into the input and remove color to restore the original. The process uses three data characteristics to generate the color: the expected value (*Ex*) depends on the data content known only to the data owner. Whereas *entropy* (*En*) and *hyperentropy* (*He*) add randomness or uncertainty which are independent of the data content and these three functions generate a collection of cloud drops to form a unique color that the providers or other cloud users cannot detect. This technique can also be applied to protect documents and images in the cloud. In cloud model, the overall property of colored drops can be represented by three numerical characters. On one hand, construct forward cloud generator to produce a lot of drops as illustrated in Algorithm 1. On the other hand, construct reverse cloud generator

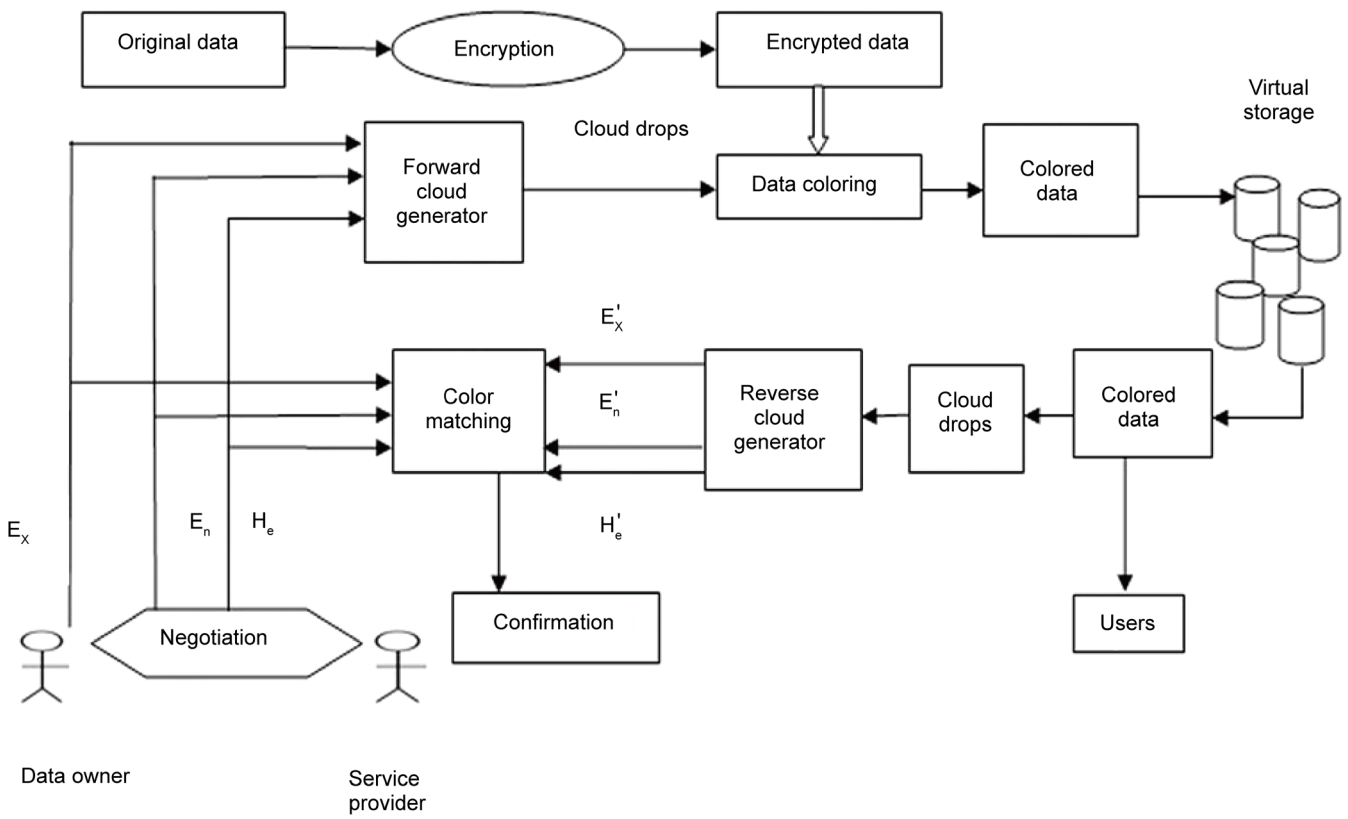

Figure 4. Block diagram for color coding generation.

to cope with the colored drops and revert *Ex*, *En*, and *He*, as illustrated in Algorithm 2.

Algorithm 1: Forward cloud generator

Step 1: Generate a normally distributed random number $En_i' = NORM\left(En, He^2\right)$.

Step 2: Generate a normally distributed random number $x_i = NORM\left(Ex, En_i'^2\right)$.

Step 3: $\mu_i = \exp\left[-\frac{\left(x_i - Ex\right)^2}{2\left(En_i'\right)^2}\right]$.

Step 4: x_i with certainty degree of μ_i is a cloud drop in the domain.

Step 5: Repeat Steps 1 to 4, and generate drops.

Algorithm 2: Reverse cloud generator

Step 1: Calculate mean $\bar{X} = \frac{1}{n}\sum_{i=1}^{n} x_i$ and variance $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{X}\right)^2$.

Step 2: $Ex' = \bar{X}$.

Step 3: $En' = \sqrt{\frac{\pi}{2}} \times \frac{1}{n}\sum_{i=1}^{n}\left|x_i - Ex\right|$.

Step 4: $He' = \sqrt{S^2 - En'^2}$.

Ex is provided by data owner; *En* and *He* are produced by negotiation of data owner and service provider. Each cloud user is provided with a value called expected value which is known only to the user. The negotiated values with the CSPs are Entropy which is unique for all users in the particular group sharing the data in the cloud. Hyperentropy is the value which is common to all the group users of the data. Then, a lot of cloud drops will be formed by forward cloud generator (see Algorithm 1) and are used to color the user data. When the data are used, the cloud drops are extracted from colored data *Ex*0, *En*0, and *He*0 will be produced by reverse cloud generator (see Algorithm 2). Final color matching which indicates data is not modified by others. Data owner and storage service provider negotiate together to select *En* and *He*, just like the key. *Ex*, *En*, and *He* are three mathematical characters. *En* and *He* can be used to transform a certain print to uncertain print drops. Figure 4 shows different paint drops according to different *En.* Also compute the entropy of each cloud drop (*En*0) and compare the difference between *En* and *En*0. To provide the continuous authentication within the group, an automated validation of data can be made at regular intervals of time. The experiment result is illustrated in the concerned tables, and the curve of case is shown in concerned Figures. The performance of the proposed system is evaluated based on the parameters viz., Execution Time, Encryption Time and Decryption Time, Memory utilization, Key size and Digital signature creation time regards with different data size. The performance results have been summarized in various tables regarding with various parameters and also conclusion has been presented. Based on the experimental results, it concluded that AES is the best performing algorithm among the various algorithms chosen for implementation with respect to encryption time and decryption time. Figure 5 describes Encryption time for various block size of data given in Table 1. It can be seen that as the block size increases the encryption time also increases gradually. Figure 6 describes Decryption time for various block size of data given in the

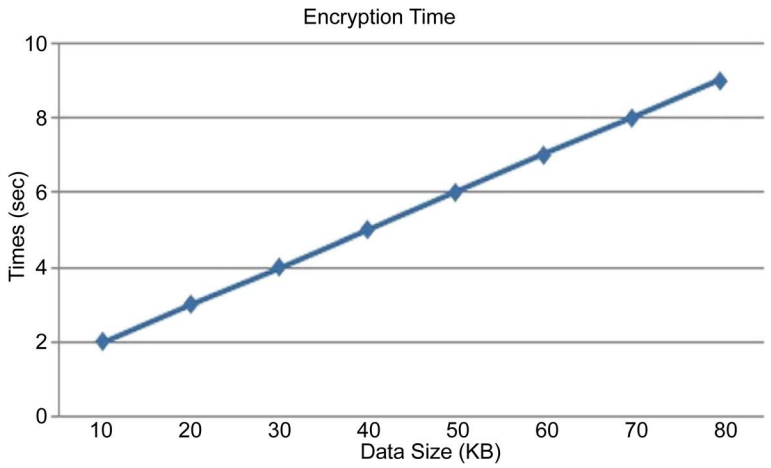

Figure 5. Encryption time.

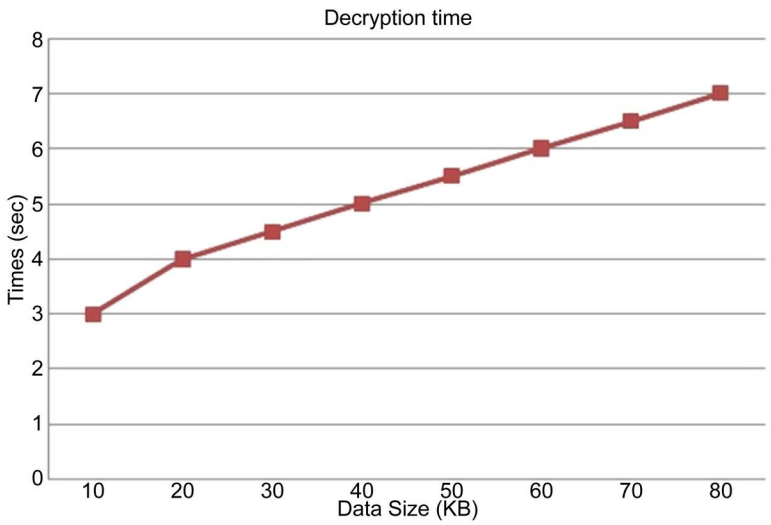

Figure 6. Decryption time.

Table 1. Encryption time.

Data size (KB)	Time (Sec)
10	2
25	3
30	4
40	5
50	6
60	7
70	8
80	9

Table 2. It can be seen from the figure that decryption time is linearly proportional to the block size. Figure 7 describes time taken for digital signature generation regards with various block size of data given in Table 3. It can be seen from the figure that the digital signature generation time is linearly proportional to the block size. Figure 8 describes time taken for executing various block size of data given in Table 4. It can be seen from the figure that the Average Finishing time is constant proportional to the block size. Figure 9 describes Resource Utilization in terms of CPU and Memory for various Data Sizes as mentioned in Table 5. Figure 10 describes Accuracy checking in terms of number of defects detected for various Data Sizes as mentioned in Table 6. Figure 11 describes Throughput in terms of Encrypted data and Time as mentioned in Table 7. Figures 12-15 gives the comparison between Two fish, Serpent algorithm with AES algorithm in terms of Encryption time and Decryption time and Execution Time, output size for each block of data given in Tables 8-11 respectively. Based on the results AES algorithm provides better performance in terms of encryption time and

Table 2. Decryption time.

Data size (KB)	Time (Sec)
10	3
20	4
30	4.5
40	5
50	5.5
60	6
70	6.5
80	7

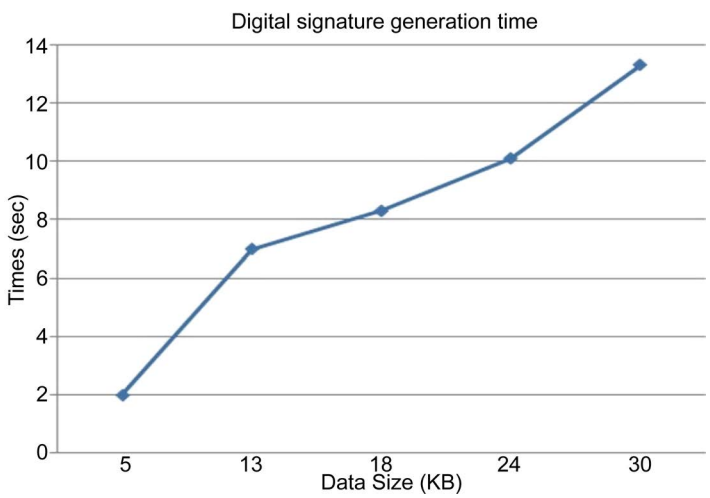

Figure 7. Generation of digital signature.

Table 3. Digital signature creation time.

Data size (KB)	Time (Sec)
5	2
13	7
18	8.3
24	10.1
30	13.3

Table 4. Execution time.

Data size (KB)	Start time (Sec)	Finish time (Sec)	Execution time (Sec) = (finish time − start time)
5	5	7	2
10	5	7	2
15	10	13	3
20	15	18	3
25	20	23	3
30	20	23	3
35	24	27	3
40	25	28	3
45	30	33	3
50	30	34	4

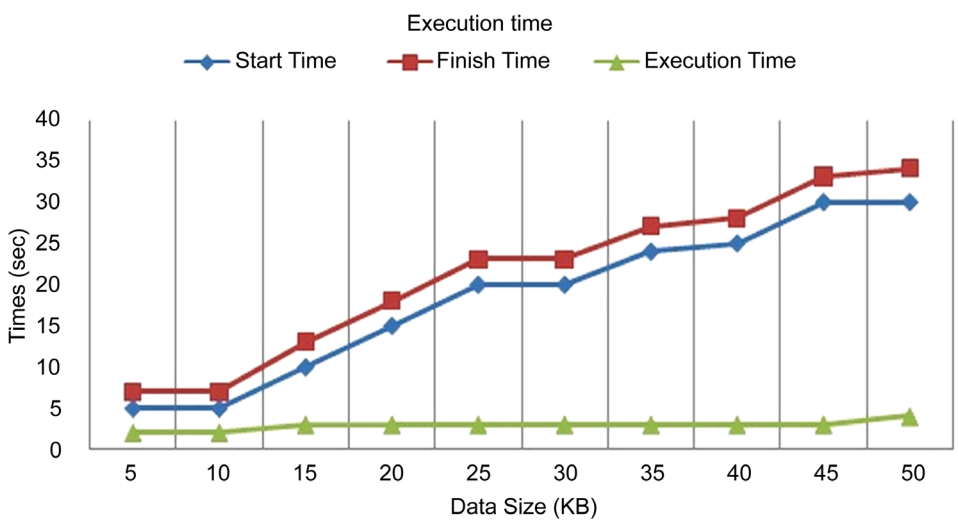

Figure 8. Execution time.

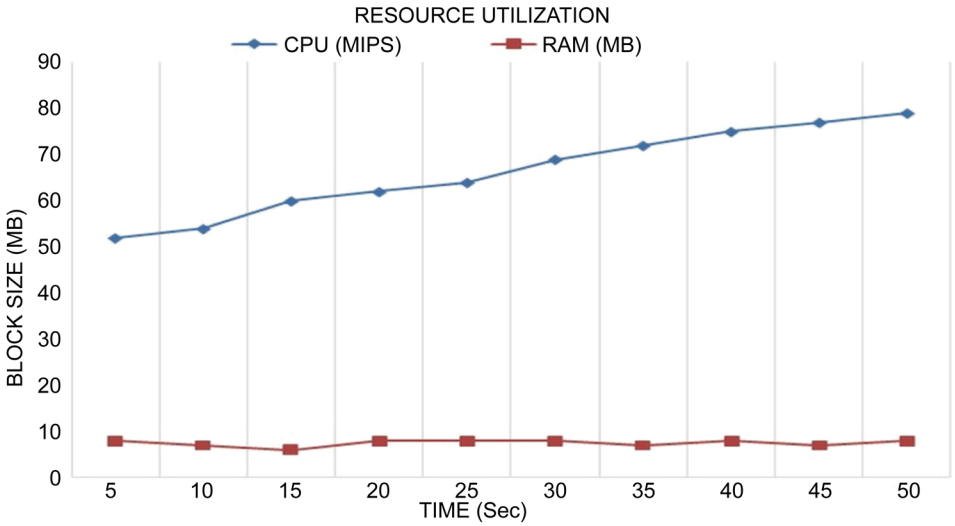

Figure 9. Memory utilization.

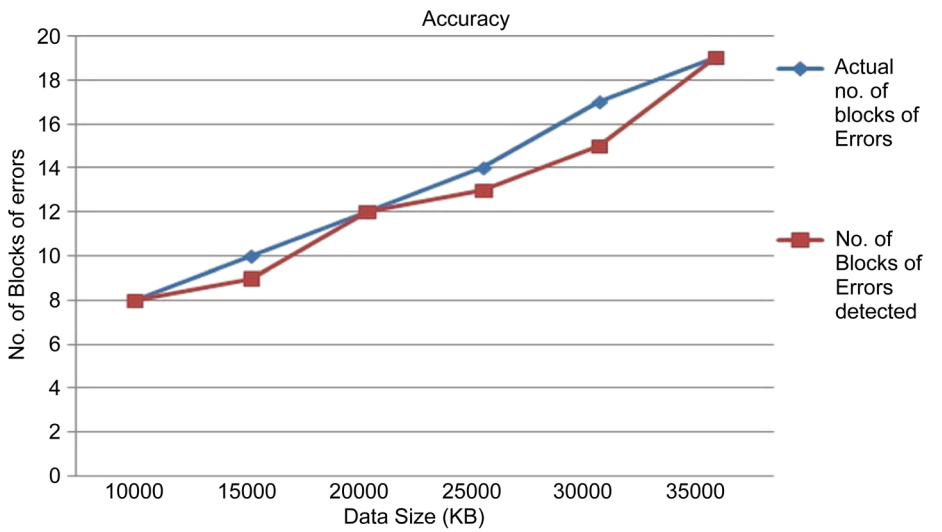

Figure 10. Detection of errors with various block size.

Table 5. Memory utilization.

Data size (MB)	Time (Sec)	CPU (MIPS)	RAM (MB)
10	5	52	8
20	10	54	7
30	15	60	6
40	20	62	8
50	25	64	8
60	30	69	8
70	35	72	7
80	40	75	8
90	45	77	7
100	50	79	8

Table 6. Accuracy checking.

Data size (bytes)	Actual No. of blocks of errors	No. of blocks of errors detected by the proposed method	Accuracy of proposed method (%)
10000	08	08	100
15000	10	09	99.91
20000	12	12	100
22000	14	13	99.91
30000	17	15	99.66
33000	19	19	100

Table 7. Throughput.

Data size (KB)	Time (Sec)	Encrypted size (KB)	Throughput
10	2	7.5	3.75
20	3	15	5
30	4	22.5	5.62
40	5	30	6
50	6	37.5	6.25
60	7	45	6.42
70	8	52.5	6.56
80	9	60	6.66

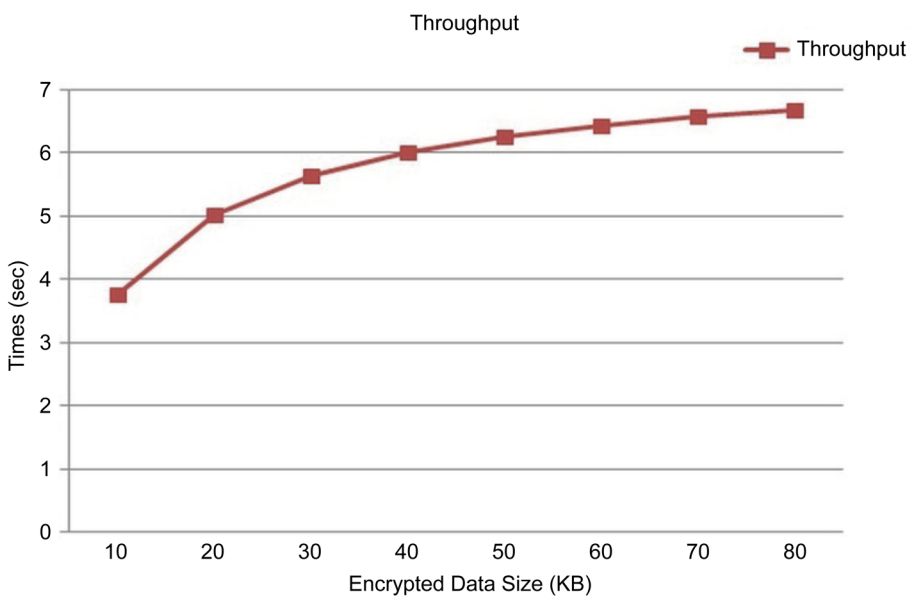

Figure 11. Encrypted data size vs time.

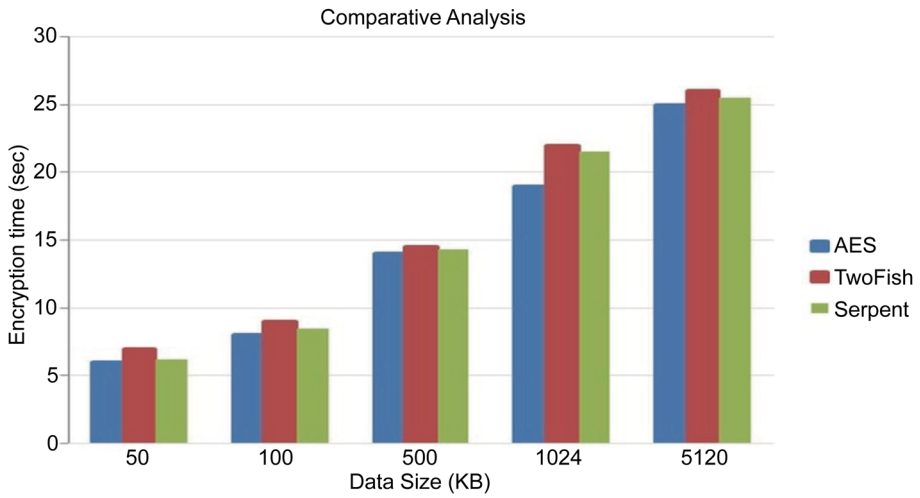

Figure 12. Encryption time.

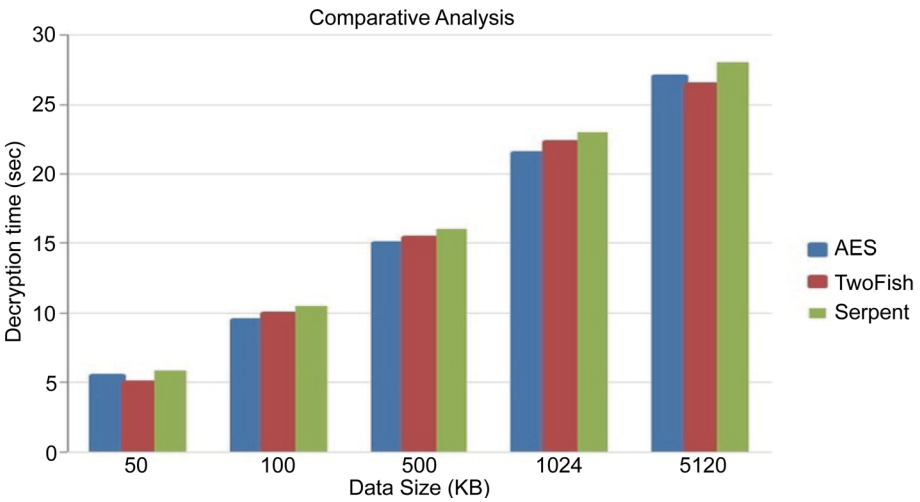

Figure 13. Decryption time.

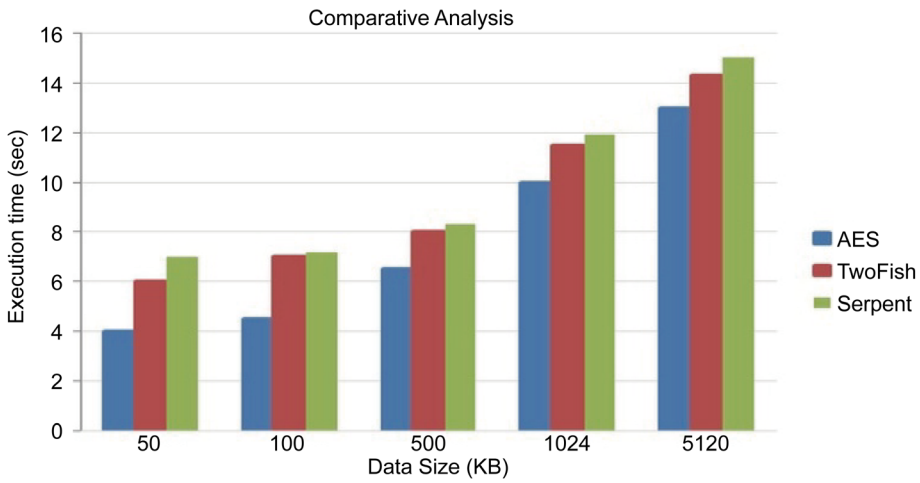

Figure 14. Execution time.

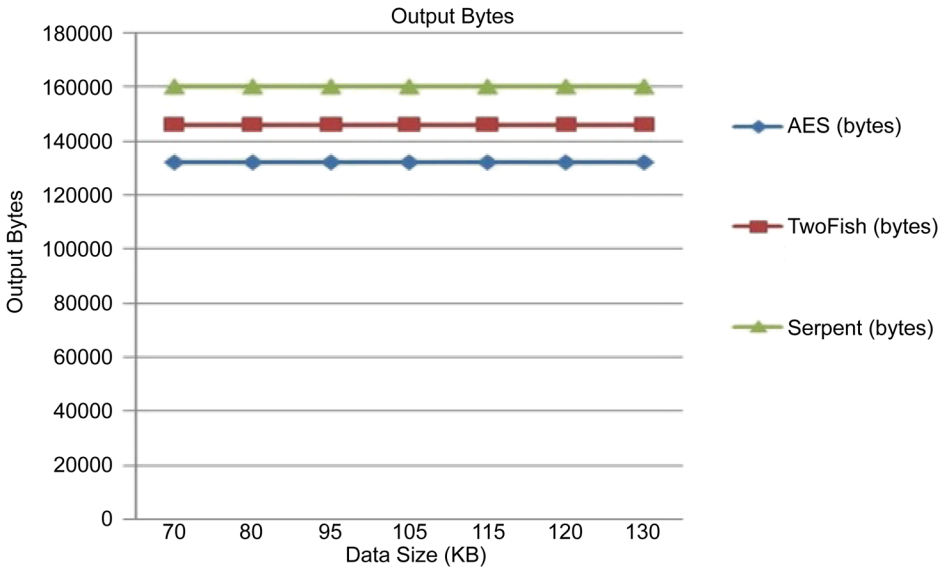

Figure 15. Output size.

Table 8. Encryption time.

Data size (KB)	AES (Sec)	Two Fish (Sec)	Serpent (Sec)
50	5.5	5	5.9
100	9.5	10	10.5
500	15	15.4	16
1024	21.5	22.3	23
5120	27	26.5	28

Table 9. Decryption time.

Data size (KB)	AES (Sec)	Two Fish (Sec)	Serpent (Sec)
50	6	7	6.2
100	8	9	8.5
500	14	14.5	14.3
1024	19	22	21.5
5120	25	26	25.5

Table 10. Execution time.

Data size (KB)	AES (Sec)	Two Fish (Sec)	Serpent (Sec)
50	4	6	7
100	4.5	7	7.2
500	6.5	8	8.3
1024	10	11.5	11.9
5120	13	14.3	15

Table 11. Ouput size.

Data size (KB)	AES (output bytes)	Two Fish (output bytes)	Serpent (output bytes)
70	132,082	146,022	160,030
80	132,082	146,022	160,030
95	132,082	146,022	160,030
105	132,082	146,022	160,030
115	132,082	146,022	160,030
120	132,082	146,022	160,030
130	132,082	146,022	160,030

decryption time and execution time. The two main characteristics of a good encryption algorithm are: Security and Speed. In this paper, analyze security V/s performance of three algorithms Two Fish, Serpent and AES based on the experimental results using cloud simulator.

3. Comparison of Results and Analysis

The performance are evaluated based on the parameters viz., Execution Time, Incryption Time, Decryption Time and Output Bytes. The encryption time is also used to calculate the throughput of an encryption scheme, calculated as the total plaintext in byes encrypted divided by the encryption time. Comparison, analysis of the results of various algorithms are performed. The Experimental result for Encryption, Decryption and Execution algorithm AES, Two fish and Serpent are shown in Tables 8-10 which shows the comparison of three algorithm AES, Two fish and Serpent using same text file for five experiment, output byte for AES, Two fish and Serpent is same for different sizes of files. By analyzing Table 11, noticed the AES has very smaller output byte compare to Two fish and Serpent algorithm. Time taken for encryption, decryption and execution by Two fish and Serpent algorithm is much higher compare to the time taken by AES algorithm. By analyzing Figures 12-14, one which shows time Taken for encryption on various size of text file by three algorithms i:e AES, Two fish and Serpent, noticed that Serpent algorithm takes much longer time compare to time taken by AES and Two fish algorithm.AES algorithm consumes least time for encryption. Two fish and Serpent algorithm shows very minor difference in time taken for encryption and decryption. Figure 15 shows the size of output byte for each algorithm used in experiment. The result shows same size of output byte for different size of text file in case of all three algorithms and noticed that Serpent algorithm output bytes are highest for all sizes of text file.

4. Conclusion

This paper presents a new technique for enhancing data security through improving data integrity violation checking over the cloud storage without using TPA. In the pro-

posed technique, the data are divided into blocks, where each block is arranged into square matrix. An element in this matrix is arranged into a new form using Matrix Dialing method which leads to memory saving through bits reduction and also to enhance accuracy of data. Also digital signature is applied on each determinant factor to enhance data integrity assurance. This model also uses data coloring on encrypted digital signature to enhance the data security which helps the user to verify and examine the data from unauthorized people who manipulate the data in the cloud storage. In this method accuracy is maintained at satisfied level by rearranging the data two times via original matrix and its corresponding Dialing method Rotational matrix. Though it requires more computation time it provides good level of accuracy and security of data. Thus, here it tries to provide a new insight to improve the cloud storage security through detection of data integrity violations in block level during storing or transmission. Encryption algorithm plays an important role in data security where encryption time, memory usages and output byte are the major issue of concern. The selected encryption AES, Two Fish and Serpent algorithms are used for performance evaluation. Based on the text files used and the experimental result it was concluded that AES algorithm consumes least encryption time and least memory usage. Serpent algorithm consumes longest encryption time and memory usage is also very high but output byte is least. The simulation results show that the new method gives better results compared to the Two Fish and Serpent algorithms and has resolved all of their deficiencies that go along with data integrity assurance methods towards data security. The performance measures viz., better encryption/decryption time and also computation time, memory utilization, and quicker detection of violation are considered. In future work this proposed model can be implemented for conducting more experiments using various algorithms and methods in cloud computing on other types of data like image, sound and multimedia data and test the performance of the proposed approach. The focus will improve encryption time and less memory usage.

References

[1] Chavan, A. (2014) Cloud Computing. *Asian Journal of Management Sciences*, **2**, 1-6.

[2] Diffie, W. and Hellman, M.E. (1976) New Directions in Cryptography. *IEEE Transactions on Information Theory*, **22**, 644-654. http://dx.doi.org/10.1109/TIT.1976.1055638

[3] Kahate, A. (2008) Cryptography and Network Security. Tata McGraw-Hill Publishing Company, New Delhi.

[4] Shantala, C.P. and Kumar, A. (2014) Integrity Check Mechanism in Cloud Using SHA-512 Algorithm. *International Journal of Engineering and Computer Science*, **3**, 6033-6037.

[5] Wang, C., Wang, Q. and Ren, K. (2009) Ensuring Data Storage Security in Cloud Computing. 17*th International Workshop on Quality of Service* (*IWQoS*), IEEE Conference Publication.

[6] Govinda, K., Gurunathprasad, V. and Sathishkumar, H. (2012) Third Party Auditing for Secure Data Storage in Cloud through Digital Signature Using RSA. *International Journal of Advanced Scientific and Technical Research*, **4**.

[7] Bhagat, A. and Sahu, R.K. (2013) Cloud Data Security While Using Third Party Auditor.

International Journal of Computer Applications, **70**.

[8] Ghaeb, J.A., Smadi, M.A. and Chebil, J. (2010) A High Performance Data Integrity Assurance Based on the Determinant Technique. Elsevier, April.

[9] Zhang, T.N.T. (2009) A Study of DES and Blowfish Encryption Algorithm. *Tencon IEEE Conference*.

[10] (2015) DES Algorithm. http://orlingrabbe.com/des.htm

[11] Coppersmith, D. (1994) The Data Encryption Standard (DES) and Its Strength against Attacks. *IBM Journal of Research and Development*, **38**, 243-250. http://dx.doi.org/10.1147/rd.383.0243

[12] Seth, S.M. and Mishra, R. (2011) Comparitive Analysis of Encryption Algorithms for Data Communication. *IJCST*, **2**, 292-294.

[13] Stallings, W. (2006) Cryptography and Network Security. 4th Edition, Pearson Prentice Hall.

[14] Singh, G., Singla, A.K. and Sandha, K.S. (2011) Throughput Analysis of Various Encryption Algorithms. *IJCST*, **2**, 527-529.

[15] (2012) Performance Analysis of AES and BLOWFISH Algorithms. *National Conference on Computer Communication & Informatics*, School of Computer Science, RVS College of Arts and Science, 7 March 2012.

[16] Schneier, B. (1994) Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish) Fast Software Encryption. *Cambridge Security Workshop Proceedings*, Springer-Verlag, December 1993, 191-204. http://dx.doi.org/10.1007/3-540-58108-1_24

[17] (2015) Blowfish Algorithm. http://pocketbrief.net/related/BlowfishfEncryption.pdf

[18] (2015) Blowfish Algorithm. http://www.schneier.com/blowfish.html

[19] Schneier, B. (2008) The Blowfish Encryption Algorithm.

[20] Camara, L., Li, J., Li, R. and Kagorora, F. (2014) Block-Based Scheme for Database Integrity Verification. *International Journal of Security and Its Applications*, **8**, 25-40. http://dx.doi.org/10.14257/ijsia.2014.8.6.03