ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

A Trust Management System for Secure Sharing of Data in Cloud Storage

Sumiti Joshi, Shiv Dubey

M.Tech Scholar, Dept. of C.S.E., AITR, RGPV University, Indore, M.P, India

Assistant Professor, Dept. of C.S.E., AITR, RGPV University, Indore, M.P, India

ABSTRACT: Nowadays every individual and organization required their digitized data in an on-demand manner. Therefore, everyone begins expanding their data on the hosting servers so that they can access the information whenever necessary. In addition, of that to reduce the maintenance cost of data management and also to secure data for a long time, the cloud servers outsource data to other third party servers. In order to preserve data on the third party server cryptographic techniques are beneficial. But during the retrieval of data and in order to regulate the data access mechanism, improved trust management approach is needed between two data exchanging parties.

Thus, using data preserving and trust management techniques a new system has been developed. The given methodology includes the implementation of data storage services and their sharing mechanism for outsourcing and offshoring process during data exchange. In further the technique is extended with the development of a digital envelope using AES and a version of SHA-2 algorithms, i.e., SHA-256 algorithm to secure data in network and storage during data sharing and exchange. In order to keep track, regarding the trust of the connecting parties, a trust evaluation technique is also associated with the system. The trust evaluation of the data accessing system is computed using the weighted method on the basis of three factors, that is Time, User rating and History behavior.

Finally, the implementation of the work is performed using JAVA technology. In addition to that, after the implementation and analysis of the system, to ensure the authenticity of the work the performance of the cryptographic system is evaluated and compared to the traditional hybrid cryptographic system in terms of time and memory. Furthermore, the service of Red Hat's Platform-as-a-service (PaaS) namely Open Shift is also used for deploying the given implementation of public clouds. The tentative results show the effectiveness and efficient methodology for data exchange with multi-party trusted environment.

KEYWORDS: Cloud Storage; Data Outsourcing; Security; Trust Management; Digital Envelope

I. Introduction

In this age of the internet, the vast majority of the applications have become online. These applications serve the data and services to the end users in an uninterrupted manner. Thus, each and every fraction of seconds a huge amount of requests is generated to find data, logic or any services. In order to handle such huge request traditional computing becomes out-dated and distributed computing takes place. These are the huge computational and storage infrastructure that supports the frequent changing data and requests. To manage and diminish the complexity of data storage on the local servers the cloud providers outsource their data to other third party servers.

The outsourcing of data needs some techniques to improve the security, transparency and trust to host the data and access the stored data on demand. In this presented work the data outsourcing concept and sharing of data in an efficient and trusted manner are investigated. Additionally, a new concept which improves the data access, sharing and storage is demonstrated. Thus, for the modern scenario, there is a need for a new security technique along with the trust assurances to keep information secure from unauthorized hosts. Therefore, the following challenges are needed to be dealt with during the implementation of the presented work:

- Identity Management
- Privacy Management
- Trust Management
- Data Exchange Security and Data Redundancy Management

DOI: 10.15680/IJIRCCE.2016. 0404028

II. RELATED WORK

In [1] authors outsourced the Personal Health Record (PHR) to a third-party cloud provider where privacy is a primary concern. Thus, they propose a new patient-centric architecture for data access control in semi-trusted servers by using Attribute-Based Encryption (ABE) techniques to encrypt data which focuses on the multiple data owner and security domains that diminish key management complexity for owners and clients along with an efficient on-demand client or attribute repudiation and break-glass access under emergency situations. In [2] authors consider a multi-factors feature of trust to propose an access control model which consists of multi-factors trust computation, a feedback module, and permission mapping which is appropriate for access control in dynamic situations. In [3] authors give an overall security viewpoint of Cloud computing with the mean to highlight the security worries that should be systematically addressed and figured out how to understand the maximum capacity of Cloud computing. In [4] authors propose a multi-faceted Trust Management (TM) framework for building design for a cloud computing commercial center to recognize the trustworthy cloud suppliers in terms of different traits (e.g., Security, performance, consistency). In [5] authors introduce a trust administration, architecture which comprises of a Cloud System Registry and Discovery that is cloud supplier's registry and records their particular trust values, a trust calculator that computes CSP's trust in view of inputs of two parameters to be specific SLA and QoS. In [6] authors explored the properties of trust, proposed goals of the IOT trust administration, and give an overview of the present writing advances towards reliable it. Besides, it examines unsolved issues, determine research challenges and demonstrate future exploration patterns by proposing an examination model for all encompassing trust management in it. In [7] authors made an endeavor to plan and simulate a system in MATLAB to compute the reliability of service suppliers in light of their consistency to guarantee SLA parameters, utilizing a synthetic data set. In [8] authors utilize a trust model which measures the security, quality and registers a trust, esteem that contains different parameters that are vital measurements along which security of cloud administrations can be measured. CSA (Cloud Service Alliance) service difficulties are also utilized to evaluate the security of an administration and legitimacy of the model. In [9] authors endeavor to call attention to different procedures to settle the protection and security issues of the information out in the public auditing scheme in a cloud environment.

III. PROPOSED METHODOLOGY AND DISCUSSION

A. *Algorithm Used:*

This section includes the various algorithms and a concept that is involved in the designing of a trust based secure data sharing model.

AES Algorithm

The acronym AES stands for Advanced Encryption Standard (AES) which is a symmetric encryption created by Joan Daemen and Vincent Rijmen. AES is most commonly utilized encryption algorithm today, which depends on a few permutations, substitutions and linear transformation, each executed on data squares of 16 bytes. Starting today, no feasible attack against AES exists. Along these lines, AES remains the favored encryption standard for governments, banks and high-security frameworks around the globe. It is valuable when we need to encode a private content into an unreadable format, for instance, when we have to send sensitive information in an email. The decryption of the encoded content has been conceivable just on the off chance that we know the right password.

The time needed to break an encryption algorithm is straight-identified by the length of the key used to secure the communication. AES permits choosing a different kind of bits such as the 128-bit, 192-bit or 256-bit key, making it rapidly more powerful than the 56-bit key of DES. AES is an iterative as opposed to a Feistel cipher. It depends on substitution–permutation system. It contains a progression of connected operations, some of which include replacing inputs by particular outputs (substitutions) and others include rearranging bits around (permutations).

The First Step:

- Add Round Key

The Following four functions are repeated periodically:

- Sub Byte
- Shift Row
- Mix Column
- Add Round Key

Final Step:

- Sub Byte
- Shift Row
- Add Round Key [10]

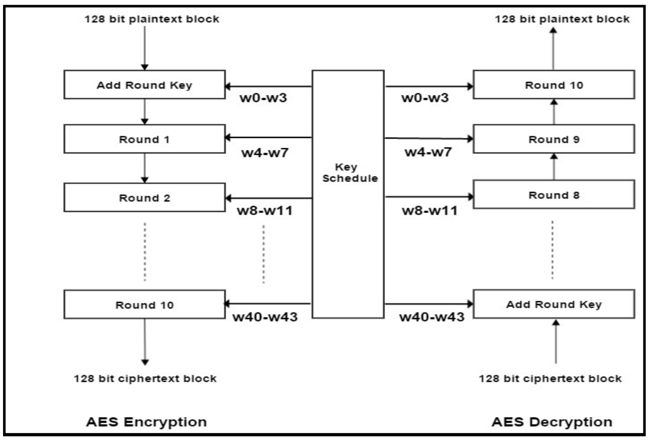


Fig. 1. The overall structure of AES with 128 bit key [10]

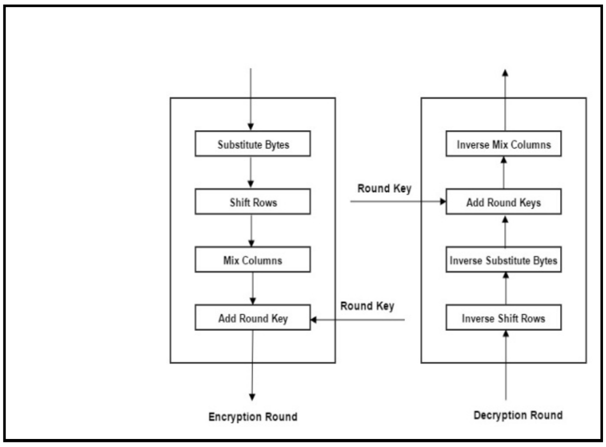


Fig. 2. One round of encryption and decryption [10]

Secure Hash Algorithm-256 (SHA-256)

SHA known as Secure Hash Algorithm (SHA) was outlined by NIST and NSA in 1993, which was modified in 1995 as SHA-1. Later SHA-2 was developed. SHA-2 is a typical name for four extra hash works additionally known as SHA-224, SHA-256, SHA-384, and SHA-512.Their suffix begins from the bit length of the message digest they deliver. The variants with length 224 and 384 are acquired by truncating the outcome from SHA-256 and SHA-512 separately. SHA-256 uses a block size of 512 bits and repeats 64 rounds while SHA-512 uses a 1024 bit block size and has 80 rounds. The SHA-2 algorithms take over the same structure of message extension and iterated state update, change as SHA-1 however, both message development and state upgrade change are significantly more complex. SHA-256 uses sixty-four constants K0,.., K63 of 32 bits each and eight registers to store halfway results H0, .., H7. The capacity definitions for SHA-256 are [11]:

$$W[t] = \begin{cases} M[t], & \text{if } 0 \le t \le 15 \\ \sigma 1\,(W[t] - 2) + W[t] - 7 + \sigma 0\,(W[t] - 15) + W[t] - 16, & \text{if } 16 \le t \le 63 \end{cases}$$

With,

$$\Sigma 0(x) = x >>> 2 \oplus x >>> 13 \oplus x >>> 22$$
$$\Sigma 1(x) = x >>> 6 \oplus x >>> 11 \oplus x >>> 25$$
$$\sigma 0(x) = x >>> 7 \oplus x >>> 18 \oplus x >> 3$$
$$\sigma 1(x) = x >>> 17 \oplus x >>> 19 \oplus x >> 20$$

and

$$f[IF](e, f, g) = e \wedge f \oplus \neg e \wedge g$$
$$f[maj](a,b,c) = a \wedge b \oplus a \wedge c \oplus b \wedge c$$

SHA -256 Algorithm:

```
SHA-256 (M):
  (* Let M be the message to be hashed *)
    for each 512-bit block B in M do
    W = f [exp] (B);
      (* Initialize the registers with the constants. *)
        a = H0; b = H1; c = H2; d = H3; e = H4; f = H5; g = H6; h = H7;
        for t = 0 to 63 do
            (* Apply the 64 rounds of mixing. *)
             S1 = h +Σ1(e)+ f[IF](e,f,g)+ K[t ]+ W[t];
        S2 = Σ0(a)+ f[maj](a,b,c);
          h = g; g = f; f = e; e = d + S1; d = c; c = b; b = a; a = S1 + S2;
        (* After all the rounds, save the values in preparation of the next data block. *)
      H0 = a + H0; H1 = b + H1; H2 = c + H2; H3 = d + H3;
      H4 = e + H4; H5 = e + H5; H6 = e + H6; H7 = e + H7;
    (* After all, 512-bit blocks have been processed, return the hash. *)
  return concat(H0, H1, H2, H3, H4, H5, H6, H7); [11]
```

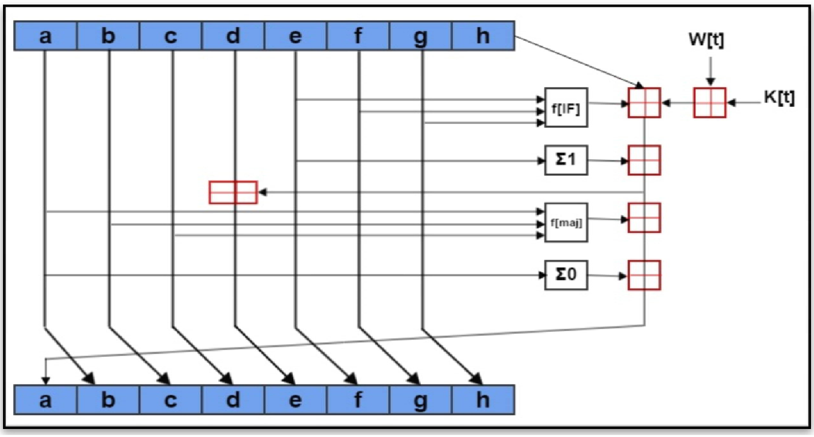


Fig.3. Single SHA-256 Iteration [11]

B. ***Proposed Methodology:***

In order to solve the above challenges and issues related to trust and security, an end to end system for managing and sharing data in cloud storage has been developed. Therefore, in the first step, a data outsourcing system is developed according to fig. 4. The given system development is divided into two modules first the primary server that has the data storage capability and the secondary server that utilizes the storage from the primary server. The involved data exchange process and access of data provide the demonstration of security and identity management. The involved components of the system are described as:

- **Primary Server:** It is used as storage service provider which provides the hosting service for the outsourced data. Additionally, it implements the data storage, off sourcing and outsourcing services to their clients.
- **User Management:** In primary server the user or data owner hosts their own personal data; therefore, it needs to create their membership with the server. According to their membership policy user can host their data.
- **User Data Manager:** It is a personalized service provided to the client where the user hosts their data and keeps on track their data according to their needs.
- **Utility Manager:** A utility manager supports the data upload, download, share and exchange data to the third party under the directions of the data owner.
- **Secondary Server:** In order to simulate the security management the secondary server is also established with the similar functions. The users of the secondary server, send a request to the primary server for data storage and on-demand data access.
- **Data Requirement:** The data owner's need their own data to access therefore some processes are implemented to upload, download, and sharing of the data hosted on the primary server. During this process security technique is initiated to manage the personalization of data, trust computation, and channel security.
- **Data Exchange:** In order to keep a track, the security and privacy management for data exchange service among both the parties (i.e. A primary server and secondary server), is implemented that contains two sub-components in the system.
- **Trust manager:** This component is used to compute the trust among both the parties, if the computed trust value found, adaptable then access to the system or data is provided to exchange the data.

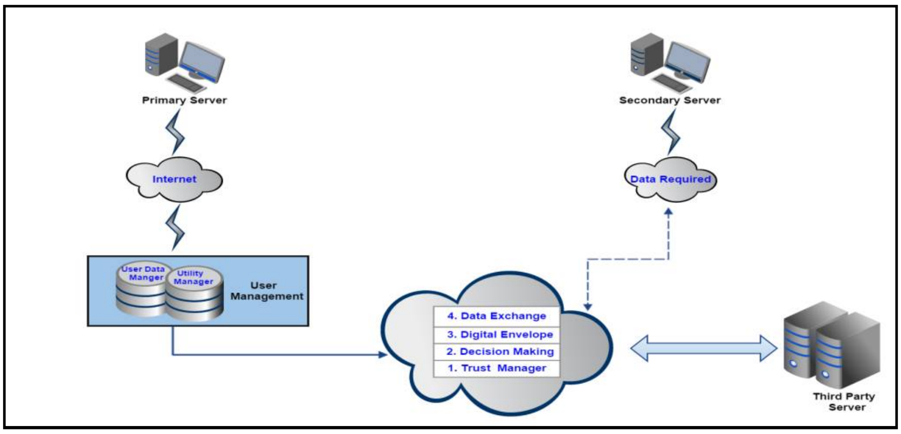


Fig. 4. Proposed Methodology

Proposed Digital Envelope

Cryptography is basically classified as asymmetric key cryptography and symmetric key cryptography. Both types of approaches have their respective pros and cons, but individually both the mechanisms do not solve all the problems related to security. Thus, combining the two approaches result in the development of Digital Envelope. Digital Envelope helps to solve the security threats in cloud computing. The basic concept of a digital envelope is such that for encryption of the data, the symmetric key algorithm is used in which encryption is performed with a one-time symmetric key and the asymmetric key is used to encrypt the session key or one-time symmetric key.

Thus, using the concept of Digital envelope, in the proposed work a data envelope or either a digital envelope is created using AES algorithm and SHA-256 algorithm. The reason for using AES and SHA-256 algorithms is it is not easy to attack the data encrypted by this algorithm thus providing high security of data. To securely deliver data between both the parties in cloud storage a cryptographic system is implemented. The overview of the proposed cryptographic data exchange technique is given in fig. 5. The steps involved in the creation of Digital Envelope are:

- The implemented digital envelope first computes the key for the requested data, by the secondary server.
- In the key generation process, the data are produced by the SHA-256 algorithm that returns the 256-bit hash code.
- In the next step of this 256-bit hash code, the hash splitting process is performed such that 256 bits are further divided into two parts of 128 bits each.
- Now in this step the first part of the hash code, i.e., 128 bits are used to encrypt the data, thus, it is working as a key in the AES algorithm.
- Now the encryption of data is performed using an AES algorithm and a key of 128 bits.
- In the next phase, after encryption of data, the generated ciphertext is divided into chunks of 128 bits by a chunk generator. Since the process of trust computation takes some additional time in the system and also for the security of the data the generated ciphertext is divided into small chunks instead of sending in bulk so that it can be transmitted faster as chunks of small size will take less time.
- After the chunk generation processes, the chunks of data are XOR with the remaining 128-bit key parts generated by the SHA-256 Algorithm. XOR has been utilized to check information sent over a system, such that there have been no changes made in the transmitted data.
- In the final step, the XORed data are incorporated into a file and ready to transmit to the secondary server.

Proposed Trust Computation Model

Trust plays a very vital role for secure sharing of data in cloud storage. There have been different methodologies that have been used earlier for trust calculation. In the proposed approach, the motivation is from the concept of multi-factors trust model for the access control [2]. Thus, a new model for trust computation has been presented in this work. In this model for calculating the trust, a Normalized approach has been followed by taking 3 factors. Such as:

- Time
- User Rating
- Average Trust or History Behavior

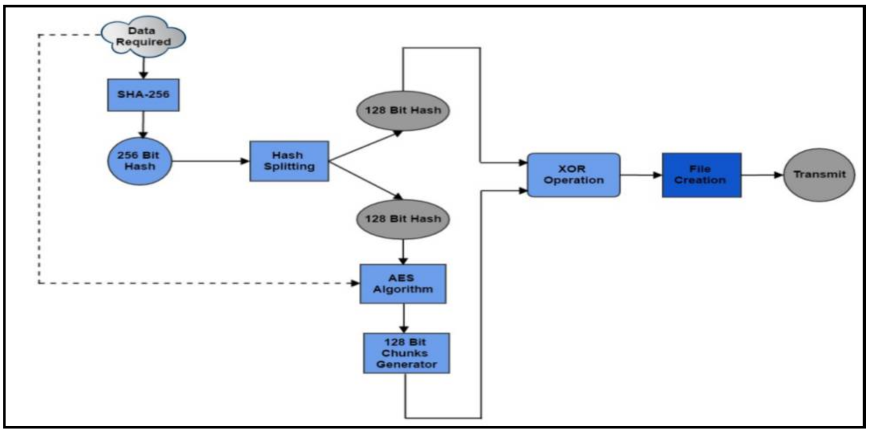


Fig. 5. Proposed Digital Envelope

Time factor depends on the time which a server took in response to the request usually in Milliseconds (ms). User rating is the rating given by the user on the basis of the performance of the server, usually value lying between 1 to 5 such that 1 for the worst and 5 for the best. History behavior indicates the average of all the trust value. Initially, Trust value has been taken as 1.

The reason for doing this is because in this model a 50% of the opportunity is given to the server to be accessible or trustworthy. It is observed that trust is inversely proportional to time, i.e., larger the server takes the time to respond lesser is the trust value. So,

$$\textbf{Trust } \alpha \ \frac{\textbf{1}}{\textbf{Time}}$$

Similarly, the trust is directly proportional to user rating and average trust, i.e., better is the user rating and average trust value more the server is trustworthy. So,

Trust α User Rating

And,

Trust α Average Trust or History Behavior

Hence, a server will be trusted only if it takes less time to respond and have a good user rating. Thus, the sum of all the 3 factors generates the Calculated Trust [Cal_Trust]. Thus, these three factors will standardize in the process of Normalization such that the values will normalize between |0-1|. The normalized value of Time, User Rating and Average Trust is denoted as Time_factor, User_rating_factor, Average_Trust respectively.

Here in the computation of trust value, 60% of Average Trust and 20% each of both Time and User rating have been considered. The reason behind giving highest weighting to past history i.e. Average Trust is it considered all the trust value obtained from the system. Thus, values for Average trust (a1), User rating (a2) and Time (a3) are 0.6, 0.2 and 0.2 respectively.The reason for this classification of values is because the sum of all the factors of the trust computation should be unity, i.e., 1 (one) [2]. Thus,

a1+a2+a3=1;

Where, **a1=0.6; a2=0.2;a3=0.2**

The normalized values have been appeared in the tables below. Thus, trust will be calculated (Cal_Trust) by the following formula:

Cal_Trust = (a1 * Average_Trust) + (a2 * User_rating_factor) + (a3 * Time_factor);

Table 1. Trust values corresponding to Normalized Average_Trust

Normalized Average_Trust/History Behavior = Average of all the previous trust calculations

Table 2. Trust values corresponding to Normalized Time_factor

Time Interval	[0-0.5]	[0.5-1]	[1-1.5]	>1.5
Normalized_Value	0.9	0.7	0.5	0.3

Table 3. Trust values corresponding to Normalized User_rating_factor

User Rating	0	1	2	3	4	5
Normalized_Value	0	0.2	0.4	0.6	0.8	1

Proposed Work

Thus, whenever data is transmitted from one server to another or shared between two servers in cloud storage security and trust management must be present. In the proposed work, this has been simulated with two servers. Suppose secondary server required some data from the primary server, so it sends a request to the primary server for the desired data. During this phase the proposed trust management system for secure sharing is initiated:

Firstly the trust manager computes the trust value with the proposed trust computation model. For example, as discussed above the values of a1, a2, a3 are 0.6, 0.2 and 0.2 respectively. Suppose an average of past trust computation is 0.59, user rating by the user is 3 and time for the response from the server are 1.7 ms then the normalized value for average trust, user rating and time are 0.59, 0.6, 0.3 respectively. So value of total trust is:

Cal_Trust = (a1 * Average_Trust) + (a2 * user_rating_factor) + (a3 * time_factor);

=(0.6 *0.59) + (0.2 *0.6) + (0.2*0.3)

= 0.354+0.12+0.06

=0.534

In the next step, the computed trust value is compared with a threshold value. A Threshold Value has been set to:

Threshold_Value= 0.5.

The threshold value is the point of trust rating, which is statically fixed to 0.5 and can vary between 0-1. So, if value of

Cal_trust >0.5

Then permission will be granted to the system to perform the activities and the data is outsourced or off sourced but if

Cal_Trust <0.5

Then the function of the system will be blocked and all its rights will be taken and an admin will recheck it. So if the server is found trustworthy, then the digital envelope is created as discussed above to secure the data required by the secondary server. Finally, the data is exchanged securely with the help of the proposed digital envelope and provided to the desired secondary server.

IV. Simulation Results

The experimental evaluation and the system performance are computed and demonstrated in this chapter. Here, the proposed work which is a combination of AES and the SHA-256 algorithm is compared with a hybrid algorithm which is a combination of DES and MD-5 algorithms. Therefore, some essential performance parameters are evaluated and listed with their obtained observations.

Encryption Time

The amount of time required to perform encryption using the selected algorithm is termed as the encryption time of the cryptosystem. The encryption time in milliseconds (ms) of the proposed and traditional system is demonstrated using fig. 6 and table 4. In this diagram the X- axis shows the different file size in KB, which is shown in table 4 on which the experiment is performed and the Y-axis shows the amount of time in milliseconds consumed for processing the input file. Additionally, the performance of the proposed system is shown using blue line and the performance of the traditional method is shown using the red line. According to the given results, the proposed system consumes less time as compared to traditional systems. Additionally, the results show the amount of time consumed depends on the amount of data provided for execution. But the respective performance of the system shows their effectiveness over the traditional system.

Decryption Time

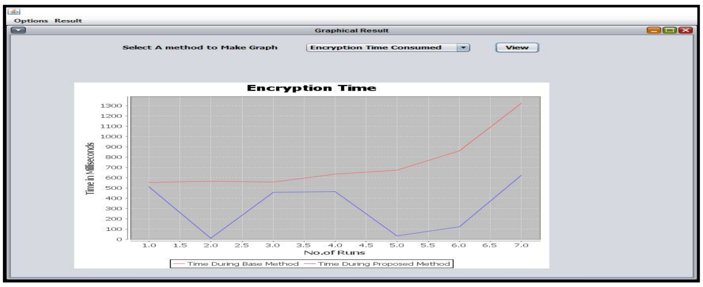
The amount of time required to recover the actual data from the ciphertext is known as the decryption time of the algorithms. The fig. 7 and table 5 shows the obtained performance of the system in terms of decryption time. The blue line shows the performance of the proposed method and the red line shows the performance of the traditional method. In the given fig. 7, X -axis shows the different file size in KB, which is shown in table 5 on which the experiments are

DOI: 10.15680/IJIRCCE.2016. 0404028

performed and the Y-axis shows the amount of time consumed in milliseconds (ms). According to the observations, the encryption time is higher than the decryption time in both the system, but the decryption time of the proposed system is much adaptable than the traditional system.

Table 4. Encryption Time Values

File Size (KB)	Proposed System Time (ms)	Traditional System Time (ms)
10	509	552
50	9	566
100	458	558
500	464	633
1000	31	674
2000	123	862
3000	621	1325

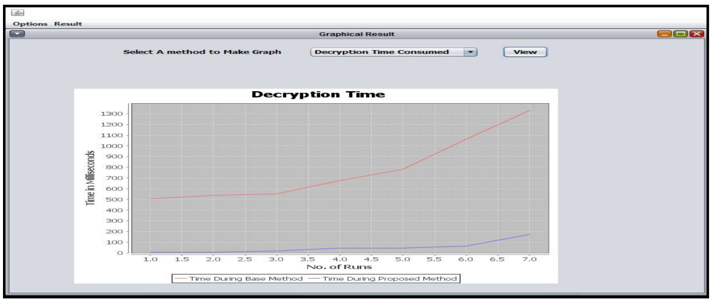


Select A method to Make Graph: Encryption Time Consumed

Fig. 6. Comparison on the basis of Encryption Time

Table 5. Decryption Time Values

File Size (KB)	Proposed System Time (ms)	Traditional System Time (ms)
10	6	507
50	5	537
100	17	552
500	45	676
1000	44	781
2000	63	1062
3000	174	1332



Select A method to Make Graph: Decryption Time Consumed

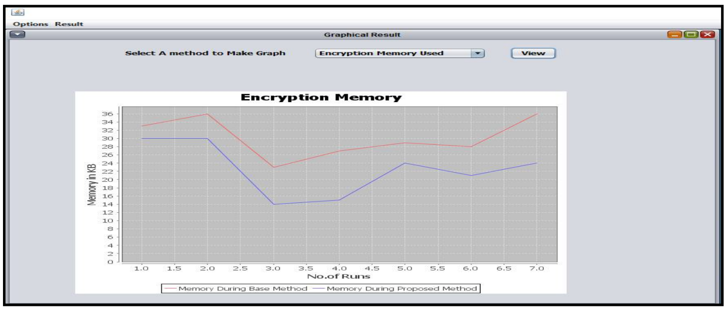
Fig. 7. Comparison on the basis of Decryption Time

Encryption Memory

The amount of main memory required to execute the algorithm with the input amount of data is known as the encryption memory. The fig. 8 and the table 6 show the encryption, memory consumption of the system. In this diagram the amount of main memory consumed in KB is shown by Y- axis and the file size, which are used for experiments are reported by X- axis. According to the obtained results, the proposed system consumes fewer resources as compared to the traditional encryption technique.

Table 6. Encryption Memory Values

File Size (KB)	Proposed System Time (ms)	Traditional System Time (ms)
10	30	33
50	30	36
100	14	23
500	15	27
1000	24	29
2000	21	28
3000	24	36



Select A method to Make Graph: Encryption Memory Used

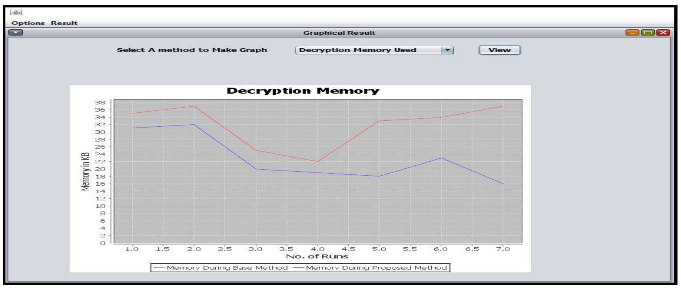
Fig. 8. Comparison on the basis of Encryption Memory

Decryption Memory

The amount of main memory required to recover the original file from the ciphertext is known as the decryption memory consumption. The fig. 9 and table 7 shows the amount of main memory in KB consumed during the data recovery. In this fig. 9 the X- axis shows the different file size in KB used for decryption and the Y -axis shows main memory consumed during the decryption. According to the obtained results, the amount of main memory used is higher in the traditional system as compared to the proposed system.

Table 7. Decryption Memory

File size (KB)	Proposed System Memory (KB)	Traditional System Memory (KB)
10	31	35
50	32	37
100	20	25
500	19	22
1000	18	33
2000	23	34
3000	16	37



Select A method to Make Graph: Decryption Memory Used

Fig. 9. Comparison on the basis of Decryption Memory

V. Conclusion and Future Work

The key objectives and aim of the work are accomplished by providing the secure and trusted environment of cloud data outsourcing. The proposed concept is adaptable with minimal resource consumption and optimum trust evaluation. In the near future, the proposed concept is extended for huge data transfer systems. Additionally, that can also be extended with the structured data trust management.

References

1. Ming Li, Shucheng Yu, Yao Zheng, Kui Ren and Wenjing Lou, ‘Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption’, *Parallel and Distributed Systems, IEEE Transactions,* Vol. 24, Issue 1, pp. 131-143, 2013.
2. Shunan Ma, Jingsha He, and Feng Gao, ‘An Access Control Model based on Multi-factors Trust’, *Journal of Networks,* Vol. 7, Issue 1, pp. 173-178, 2012.
3. Ramgovind Sumant, Eloff MM, and Smith E, ‘The Management of Security in Cloud Computing’, *Information Security for South Africa (ISSA), 2010* IEEE, pp. 1-7, 2010.
4. Sheikh Mahbub Habib, Sebastian Ries, and Max Mühlhäuser, "Towards a Trust Management System for Cloud Computing", *Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE 10th International Conference on* IEEE, pp. 933-939, 2011.
5. Monoj Kumar Muchahari and Sujeet Kumar Sinha, ‘A New Trust Management Architecture for Cloud Computing Environment’, *Cloud and Services Computing (ISCOS), 2012 International Symposium on*. IEEE, pp. 136-140, 2012.
6. Zheng Yan, Peng Zhang, and Athanasios V. Vasilakos, ‘A survey on trust management for Internet of Things’, *Journal of network and computer applications,* Vol. 42, pp. 120-134, 2014.
7. Jagpreet Sidhu, and Sarbjeet Singh, ‘Compliance based trustworthiness calculation mechanism in cloud environment’, *Procedia Computer Science,* Vol. 37, pp. 439-446, 2014.
8. Rizwana Shaikh, and M. Sasikumar, ‘Trust Model for Measuring Security Strength of Cloud Computing Service’, *Procedia Computer Science,* Vol. 45, pp. 380-389, 2015.
9. K. Selvamani, and S. Jayanthi, ‘A Review on Cloud Data Security and its Mitigation Techniques’, *Procedia Computer Science,* Vol. 48, pp. 347-352, 2015.
10. AES: The Advanced Encryption Standard URL: https://engineering.purdue.edu/KaK/compsec/NewLectures/Lecture8.pdf.
11. Cryptography in Context. URL: https://www.staff.science.uu.nl/~tel00101/liter/Books/CrypCont.pdf