Global Scientific JOURNALS

GSJ: Volume 7, Issue 12, December 2019, Online: ISSN 2320-9186

www.globalscientificjournal.com

IMPROVEMENTS TO THE CONFIDENTIALITY AND INTEGRITY OF DATA STORED IN CLOUD STORAGE

Dr. Amjad Farooq[1], Abdul Rehman[2]
University of Engineering and Technology, Department of Computer Science & Engineering
Lahore Pakistan.
Contact: amjadfarooq@uet.edu.pk, phone +92-03004174386
Contact: mscsstudent417@gmail.com, phone +92-03214507593

Abstract— Cloud computing is a new computing method, which is widely emerging technology in the recent years is used by many of the IT companies and other organizations. Cloud computing allows individuals and companies to gain access to vast computing assets without capital investment. It actually means that users can use computing assets in pay per use manner. The cost of storing huge amount of data in the local storage is burdensome than cloud storage. However, the cloud atmosphere is untrusted as it is accessed by Internet. That's why, people have security concerns on data stored in cloud environment. We intend to propose a new method for securely storing data in cloud and an integrity checking method to verify data integrity and confidentiality at the time of information retrieval.

I. INTRODUCTION

Cloud computing is a recent technology that uses the Internet, central servers to organize the data and applications, which the user can access. Cloud computing allows individual users and other business peoples to use application without the necessity to install in their computer. They can access their files, which is located in other computer using Internet. This technology allows for more inefficient computing by centralizing storage, processing memory, and bandwidth. Cloud computing comes in three categories such as Software as a Service (SaaS), Infrastructure as a service (IaaS), Platform as a Service (PaaS). The SaaS provides application software which the user can use. The Paas provides the platform for the user to do his operation. The Iaas provide physical or virtual devices for user.

As cloud computing is popular and in demand similarly cloud storage technology has greater demand. Cloud storage is a virtualized storage area over a network basis. It provides services on the basis of QoS assured. Cloud storage consist of many resources but yet act as single system. It has greater fault tolerance by redundancy. As the data generated by IT sectors are dramatically growing one can't just update the hardware frequently instead cloud storage is adopted which is a better choice. We can use cloud storage for different purpose just backing up our home desktop data into cloud storage or as an archive to maintain data for regulatory. Cloud storage allows user to access broad range of application and resources immediately, which are hosted by others. Fig 1 shows a simple cloud storage architecture.

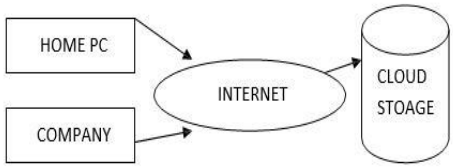


Fig. 1 Simple Cloud Storage [1]

Cloud storage architectures are mainly about delivering storage on demand in a highly scalable and multi-tenant way. Basically, cloud storage architectures contain of a front end that exports an API to communicate with the backend storage.

In traditional storage systems, this API is the SCSI protocol; nonetheless in the cloud, these are evolving protocols. At this layer, there are web service, file-based Internet SCSI or iSCSI front ends. This layer is the first communication point between the user and the service provider. Users access the services using their credentials.

The midpoint component layer is called storage controller that interconnects and communicates from the front API to the backend storages. This layer has a variety of features such as replication, traditional data placement algorithms with geographical location. Finally, the back-end consists of physical storage for data. This may be a central protocol that runs dedicated programs or a traditional back-end to the physical disks. Fig 2 shows the relationship between the aforementioned layers.

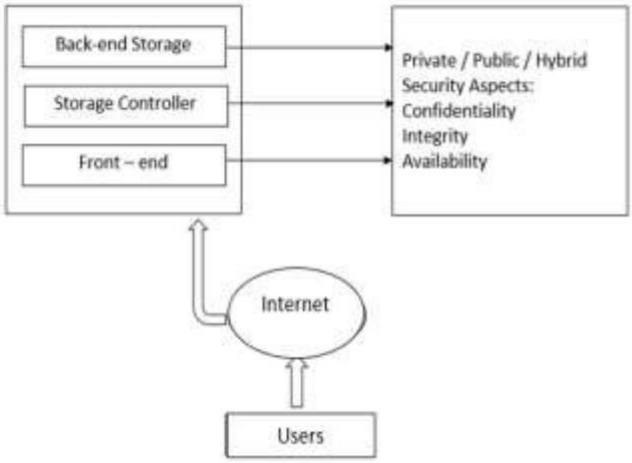


Fig. 2 Generic Cloud Storage Architecture [2]

Security is the protection of information assets through the use of technology, processes, and training. Cloud storage is a service that includes inherent vulnerabilities, but these have never discouraged users from taking advantage of its economies and flexibilities. With adoption of a cloud model, users lose control over physical security. Users raised concerns whether their data are accessed by unauthorized person since there are many user sharing the resources over the cloud.

Sharing the cloud with other users possesses risks and concerns over security. Security overall covers mainly three aspects: Confidentiality, Integrity and Availability (CIA). These aspects are the topmost considerations in designing a security measure to ensure maximum protection. Fig 3 shows the relationship between security aspects and security challenges.

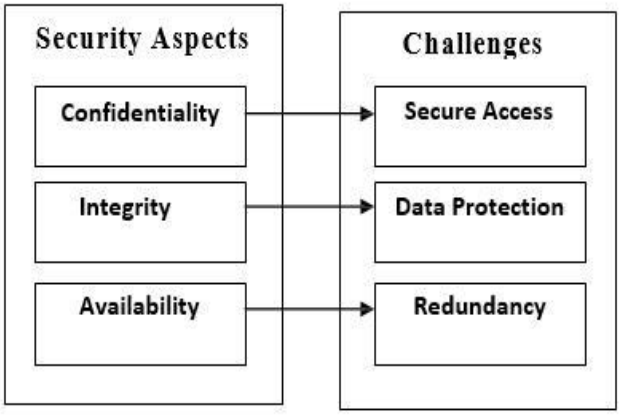


Fig.3 Cloud Storage Security Aspects and Challenges [3][4]

In this research a methodology is proposed to enhance the confidentiality and integrity of the data stored in cloud storage. The proposed method aims to handle these issues at application level, thus preventing unauthorized users from accessing information.

II. LITERATURE SURVEY

In 2018 D. Hyseni and A. Luma proposed a method for storing data in cloud storage. Proposed model deals with different scenarios depending on the data sensitivity. Model offers reliability of data by applying security mechanism at application level. Method involves usage of different cryptographic algorithms to enhance the security of data. Security depends on file encryption and file partitioning. To enhance the security, the proposed model offers three encryption strategies named as symmetric schema, asymmetric schema and hybrid schema. For measurement three symmetric cryptographic algorithms DES, TripleDES and

AES, asymmetric algorithms RSA, Deffi-Hellmen, AlGamal were used. Hybrid Schema is the mixture of symmetric and asymmetric algorithms. There could be two approaches to achieve security: partition then encrypt or encrypt then partition, author have preferred partitioned then encrypt. A file is partitioned into fixed sized blocks after that encryption is performed for each block individually by using either symmetric, asymmetric or hybrid schema. These file blocks are then uploaded to the cloud and their mapping information is stored in local cloud. From proposed encryption strategies symmetric encryption is proved to be more efficient. The method aims to enhance the confidentiality of the data. [5]

Vijayalakshmi and N. Veeraragavan focuses on the confidentiality of data by proposing a secure unified model for data confidentiality. The systems consider the cloud storage untrusted as to be owned by the third party. The communication between cloud service provider and the client is bound to application which is responsible for providing the confidentiality from untrusted user and also from the third party cloud service provider. By hiding the details of file storage (location, cryptographic algorithm used etc.) the application makes the underlying storage details vague to the user. The approach classifies data into two groups numeric data and alphanumeric data. The heart of the approach is encryption and obfuscation function. If data is alphanumeric then AES encryption is applied followed by obfuscation function. If the data is numeric then only obfuscation function is applied. The results are referred to as encrypted an obfuscated data which is then stored in cloud storage. For retrieval same sequence of operations are applied in reverse order. The resulting model has benefits over the previous work as client has no control over the data as all the storage details are hidden. Furthermore, the proposed model uses AES encryption algorithm. [6]

Geethamani and Ranjani proposed a method which consists of five modules namely registration, upload file, admin, download and key generation. Registration module use to register a particular user. Upload file module is used to upload file and its relevant meta data in encrypted form. Admin module is responsible for managing application level issues such as resolving password recovery, key distribution, maintaining and sending auditing information etc. Download module is used to download file and its meta data to verify its integrity. Key generation module is responsible for generating the key. The working of the system starts with the user registration, which is preceded by the key generation to make user able to save files in cloud storage. User can request key, password recovery and auditing information through admin module. Upload module is responsible for uploading file and its meta data to the cloud storage and archive storage (third party dedicated storage service). Download modules downloads a file and compare its meta data if it is correct file is delivered to the user. In case of modifications the file is retrieved from the archive (dedicated third party storage). [7]

A Valerian and C. Nadunagyu delivered an approach to enhance the confidentiality of cloud data storage. The methodology focuses on the development of application following a strategy to store data in cloud. The resulting algorithm starts by taking a salt value from the user. The target file is encrypted using the salt-key mixture. Where the key is generated randomly for the encryption algorithms. By generating separate key-salt pair for each file randomly increases confidentiality of user data. The final cipher text is further reversed to further enhance the security of the data. Author prefers AES encryption schemes for proposed systems due to is strength and efficiency as reported by the author to be more than the other symmetric and asymmetric cryptographic schemes. [8]

R. Kulkarni and V. Waghmare proposed an entirely different technique for securing data over the cloud using behaviour profile and decoy technology. By monitoring abnormal access pattern to the cloud. In this system whenever a user tries to get a legitimate users data a decoy file viewer is created that looks the similar like the original file viewer. The system works by presenting a user with application interface. Whenever an intruder tries to reach a user's account and tries to login to the system by using hit and trial username-password combination a decoy view is generated which looks similar to the users account. For every attempt there exists a decoy view. This technique not only keeps

the intruders busy but also makes it difficult to identify the original user's account. [9]

In 2017 E. Agrawal and P. Ram proposed an encryption algorithm to enhance the security of the data stored in cloud. The technique is symmetric substitution which uses a random number. The proposed algorithm starts by reading the input text. The key value is appended in front of the file. After finding the ASCII code for each text character it is converted into its corresponding binary value which is then complemented. The complemented binary is then converted into its corresponding decimal number. The decimal value is then divided by 4 and after finding the equivalent ASCII code for quotient. Finally, quotient is merged with remainder to get the cipher text. Finally cipher text is stored in cloud.

The decryption of the proposed algorithm is slightly different than the encryption process. First the algorithm takes the cipher text and splits the two-digit cipher text into single - single digit. After multiplying first digit with 4 and adding the second digit into the multiplication result. After performing complement operation on the results the decimal value is obtained, which is then converted into its corresponding ASCII value. Added key value is removed to get the plaintext. The efficiency of the above approach was tested by implementation in C# .NET environment. Amazon s3 services were used as cloud storage. The technique proved to be efficient for symmetric encryption. [10]

In 2016 S. Ksasy and E. Takieldeen works to enhance the security of the data by offering cryptographic algorithm known as cryptobin. The algorithm starts by taking a byte from the message. After taking the input from user for key generation the following equation is applied to get the key

(Any number) MOD 8……(I)

Equation (I) generates a number from *0 to 7*. The output of the equation will be used to allocate which bit of each byte the system will swap (*1 to 0* and vice versa). Another number is used to determine the interval length. Message encryption starts by dividing each byte into two halves as shown in Fig 5. The first half of each byte remains unchanged whereas the NOT of second part is taken. The encryption is performed by merging these two parts again. The decryption is the reverse of the encryption process.

The figure 3 shows the working of the algorithm.

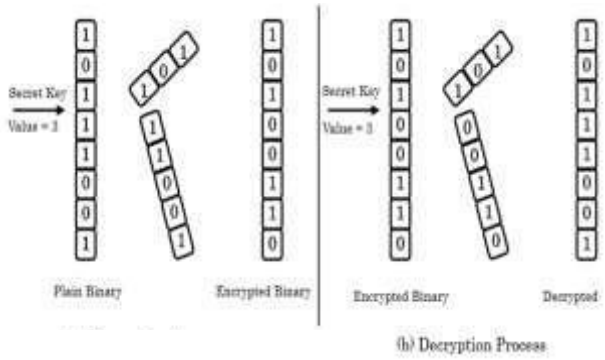


Fig. 3 Advanced Cryptographic System for Binary Codes [11]

M. Naik and P. Tungare proposed color cryptography using substitution method. The system is based on symmetric encryption which is based on encrypting text into colour images. Each character of the message is encrypted into a colour block. The user enters a message which is the actual plaintext. The channel is used from three colours red, green and blue(RGB). All the characters are converted into colour blocks. Each character present in the plaintext is replaced with one decillions colours in the world. This technique prevents from birthday paradox, meet-in-the-middle and brute force attack. Fig 4 shows the working of colour cryptography.

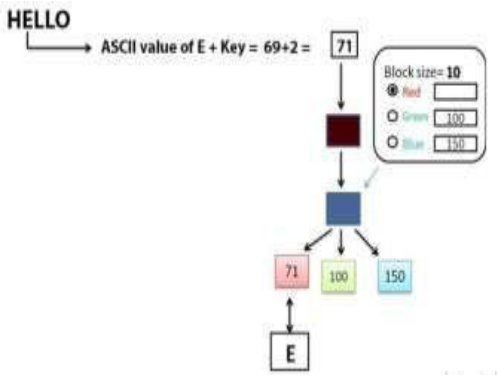


Fig. 4 Working of Colour Cryptography [12]

P. Sonia and S. Grewal proposed a methodology to validate the integrity of the transmitted data. At the receiving end the message is converted into cryptographic hash code by using SHA-1 or SHA-256. The hashing code length (24, 32, 40 and 64 bits) is taken. The hash code is converted into polynomial function by using two variable hashing function as shown in equation (II).

$$f(x,y) = \sum_{0 \le i \le d_x, 0 \le j \le d_y} A_{ij} x^i y^j \quad \ldots\ldots\ldots\ldots (II)$$

At the end polynomial based code is transmitted to the receiver. At the receiving end the message is retrieved in original plaintext form using polynomial verification method. Each receiving systems is assigned an identification number which authorize message reception. The proposed method not only provides security but also provides the comparative analysis SHA-1 and SHA-256.Further comparative analysis can be performed on the basis of processing gain, delivery ratio, energy consumed and duty cycle.

Processing gain is the amount of time it takes to convert a message from normal form to the polynomial form. Delivery ratio is the probability of message successful transmission and reception. Energy consumed in converting a message from normal form to the polynomial form. Duty cycle is the ratio between the time it takes to convert a message over the total time takes the process. After a keen analysis SHA-256 is found to be more efficient. [13]

By the same year J. Puranik and A. Giri works on two tear model of cloud, which consists of client and server. A semaphore based solution is proposed which allows data sharing in multi- user environment. When a client requests a server authentication is invoked at both ends. A semaphore is set at server side for each unique user id. Fig 5 Shows the working mechanism of semaphores.

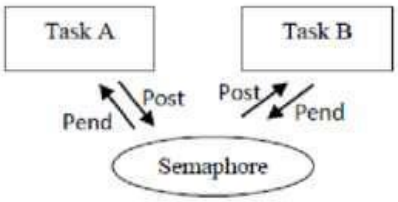


Fig 5 Working of Semaphore

A semaphore could be a variable or an abstract data type which is used by multi user operating system to provide controlled access to the shared data. A semaphore has a lock facility which is used to provide only single access to a shared resource. When one user acquires the lock other cannot access that particular shared data, whereas on releasing the lock that shared piece of data can be accessed by other users by acquiring the lock through semaphore. [14]

S. Banu and N. Deepa proposed a time based sharing scheme in cloud.in traditional client server module, an owner sends data to the cloud, the clients can get that data any time anywhere. This may result in the wastage of storage space and can cause security threats because of all time availability of data. In Data Self-Slaughter technique an owner only sends data when it Is requested by the client. While uploading the data owner can specify the time of uploading and time when that particular data can be accessed. After the completion of time data is self-destructed at cloud side. This method not only save the precious cloud storage space but also provide secure access to cloud through application.

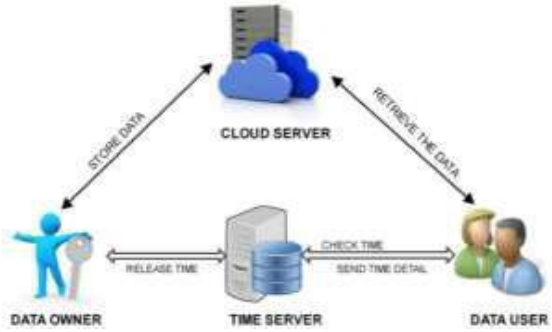


Fig 6 Working of Protected Data Self-Slaughter [15]

III. PROPOSED METHODOLOGY

A. High Level Working of Application

1. User sends request to TPA (Permission Manager) to atomically gain the access to the file.

2. After getting permission user can send the request to the cloud storage provider to upload and download file.

3. At a given time, a file can be manipulated by or can be the property of only one user.

4. After updating file user again release the lock on file through TPA (Permission Manager) making it available for other users.

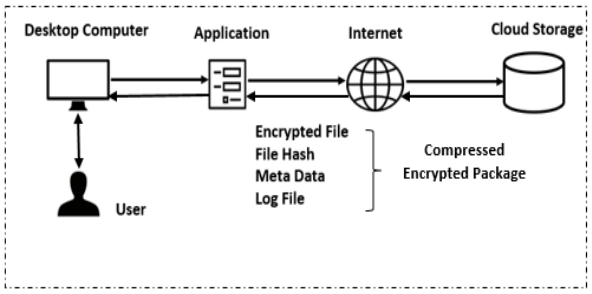


Fig. 7 High Level Working of Application

B. Storing File, File Meta-Data and Hash of File

1. Against each data file say F a log file LF is generate. The purpose of log file is to keep track of the users who have accessed data, which is used later for auditing purpose.
2. The compression and encryption is performed on data file F using 7zip compression and AES-128 to reduce the size of the data file and to put the data into a form that cannot be easily understand.
3. The hash of the data file F is generated using SHA-256 to later validate the integrity of the data.
4. A meta- file is generated which stores:
   - Data File Name (to hide the data file name).
   - File Size (used later to check the file originality or any modification).
   - Last Modified Date (to catch any unauthorized modification).
5. Meta-file is compressed and encrypted to reduce the size of its contents and for data unpredictability.
6. Using SHA-256 hash of the meta-file is generated which is later used to check the integrity.
7. In the same way the compressed encrypted version of log file LF is generated using 7zip and AES-128.
8. Hash of the log file is generated using SHA-256 The corresponding six files generated against a data file are uploaded to the cloud storage in packaged form. Fig 8 shows the working of methodology.

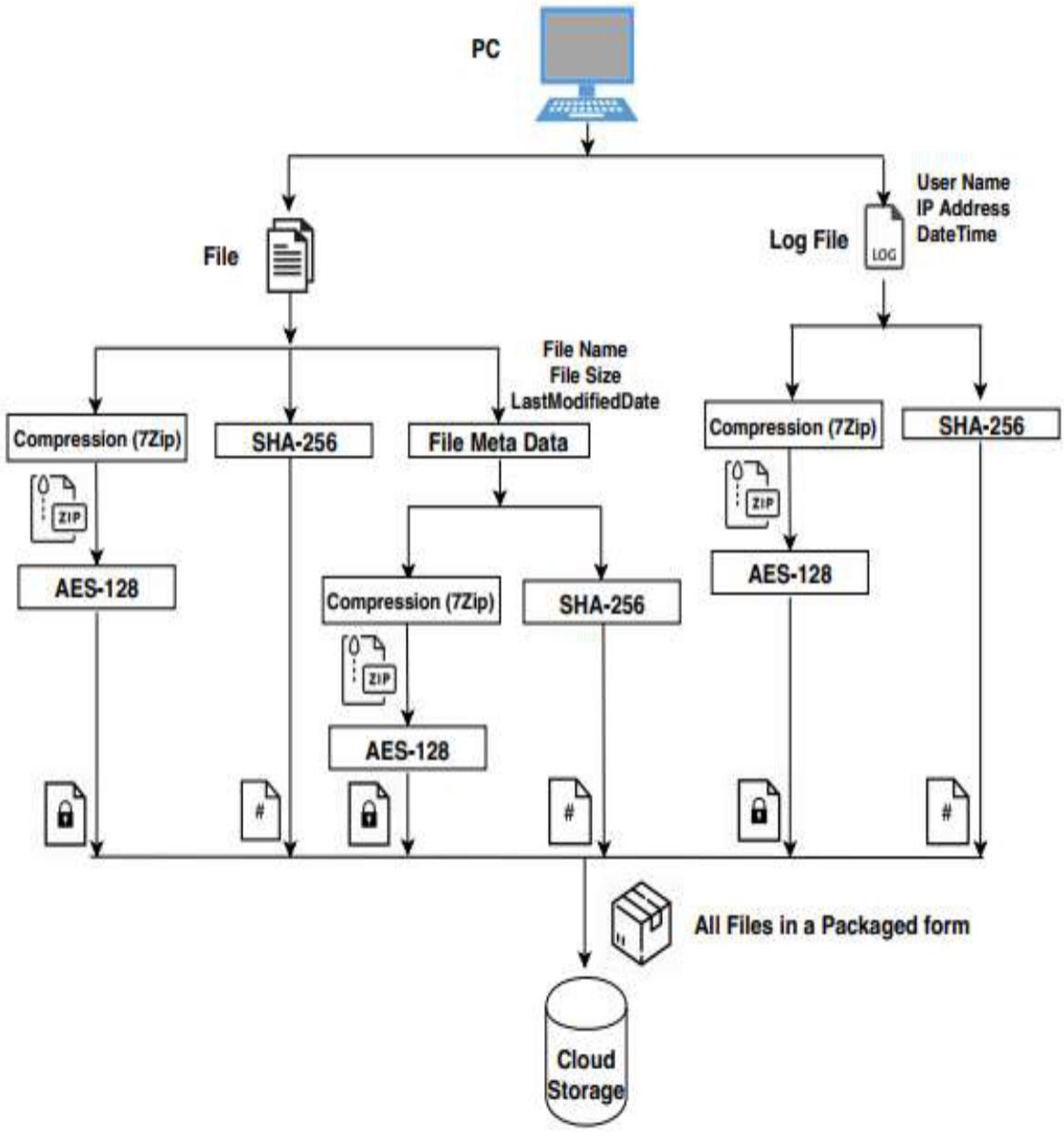


Fig. 8 Uploading File in Cloud Storage

C. Retrieving File, File Meta-Data and Hash of File

1. Download the file package from cloud storage.
2. Decrypt and Un-compress data file, meta-file and log file.
3. After generating hash of meta-file using SHA-256 compare it with the stored hash of meta file.
4. Generate the hash of data file F and compare it with the stored hash.
5. Compare data file with meta-file contents (File Name, File Size, Last Modified Date).
6. Generate the hash of log file and compare it with the stored hash.
7. If steps 1-7 succeeds file integrity is preserved.
8. In case of failure at any step send a request to the admin for file and stop.

Fig 9 shows the flow of file retrieval

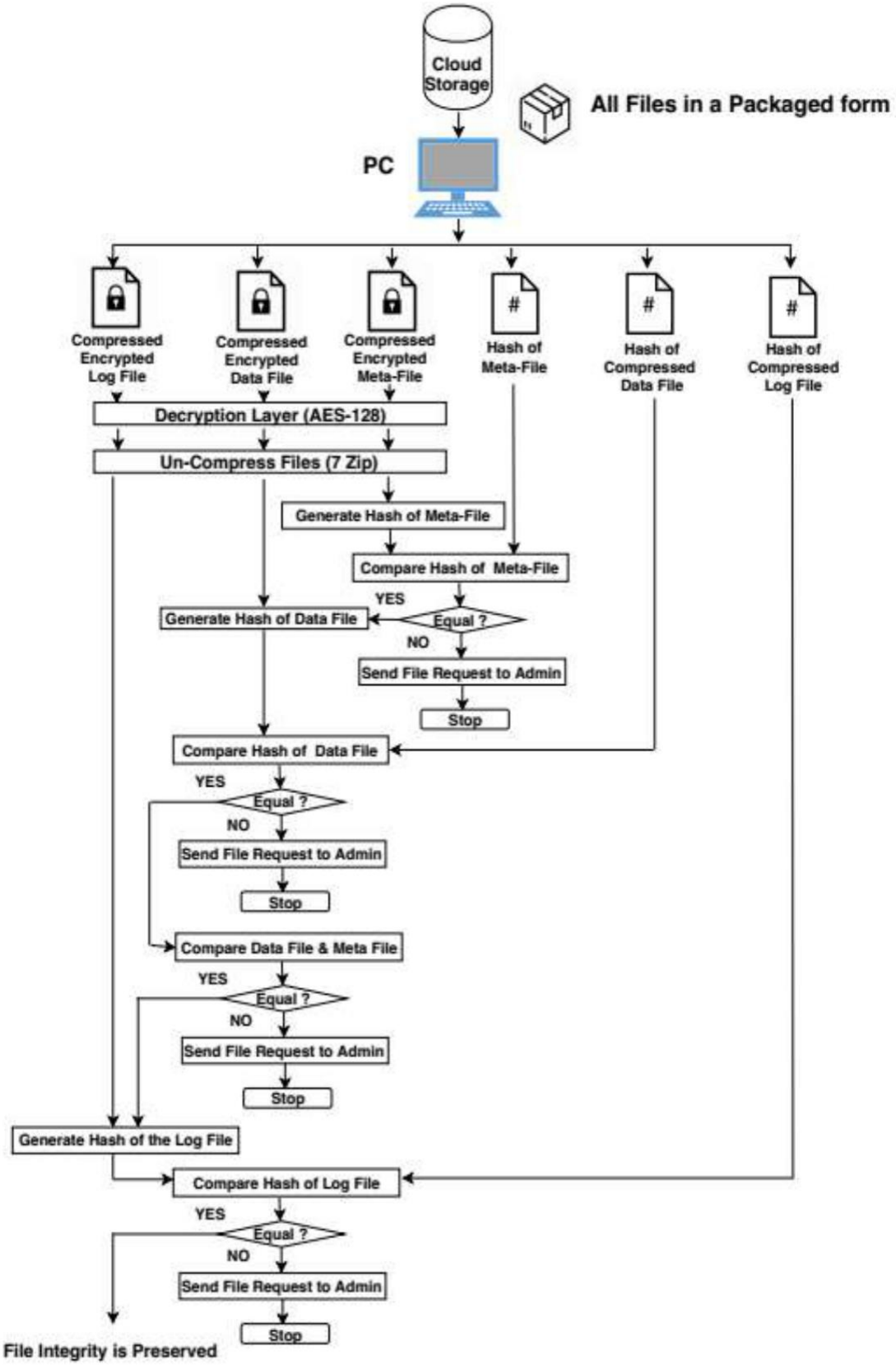


Fig 9. File Retrieval from Drop Box Storage

A. Experimental Setup

The proposed methodology has been implemented by using drop box. There are many experiments that have been carried out in past on this machine which have 8GB DDR4 RAM Intel(R) Core(TM) i7-2120M processor. Internet connection used is Wi-Tribe 4G LTE Advanced. The machine is armed with Windows 10 Pro 64-bit operating system. The software which is used for the development purposes is Visual Studio 2015. The implementation is done in C# programming language.

Dropbox is a simple storage service provider over internet, where we can store our data via internet. Dropbox provides humble interface using which we can upload and retrieve our file at anytime, anywhere. Dropbox provides reliable, highly scalable, fast and cheap data storage substructure to the users. It provides free trial base access to the developers, after that they can renew it.

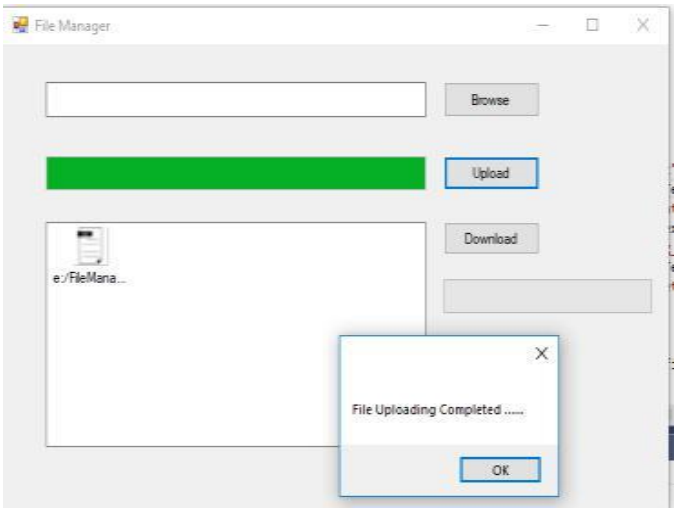


Fig. 10 Uploading File to Cloud Storage

Important Concepts of Drop Box

APP: APP is a logical connection between dropbox storage and your application.

AppFolder: Use to connect an application to specific folder (AppFolder). An AppFolder is an object of container which is stored in Dropbox. images/imagename.jpeg is stored in the AppFolder, then it is addressable using the URL:http://AppFolder.dropBox.dropboxws.com/images/file.txt.

Object: Objects are the entities (.txt, xlxs. docx, jpg) etc. which are stored in Dropbox

APPKey: A key for an APP has a unique identifier for APPFolder.

Regions: Data stored in drop box is visible to all regions you need not to worry about settings like in other service providers Amazon or 4Shared we need to specify the regions explicitly etc.

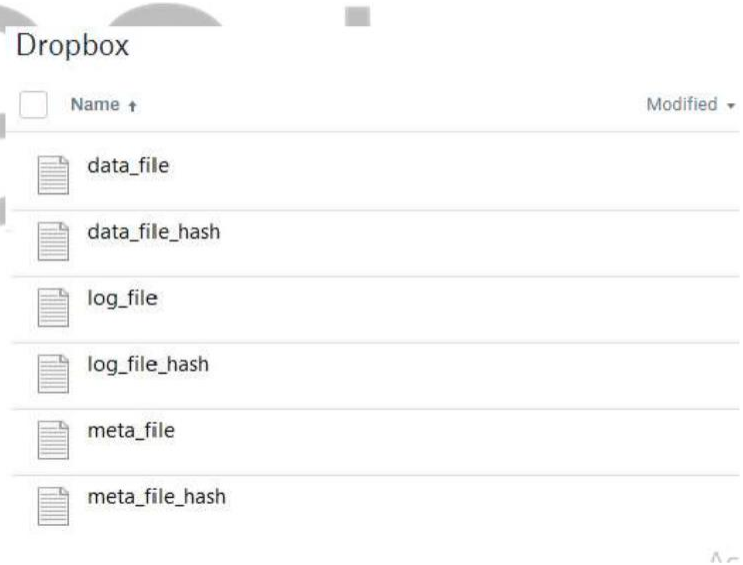


Fig.11 Corresponding Six Files in Cloud Storage

Fig 10 and Fig 11 shows the file uploading using the proposed methodology and the six corresponding files in the cloud storage respectively.

IV. RESULT AND DISCUSSIONS

A. RESULTS

File Size	MD4 (Millis)	MD5 (Millis)	SHA-256 (Millis)	SHA-512 (Millis)	SHA-192 (Milli)
1 MB	2.01	2.00	1.32	1.40	1.47
2 MB	2.35	2.40	1.41	1.50	1.51
3 MB	2.39	2.60	1.59	1.60	1.65
4 MB	2.41	2.90	1.71	1.80	1.85
5 MB	2.52	3.00	1.92	2.00	2.02

Table 1 Time to Calculate Hash for Different File Sizes with Different Hashing Algorithms

From Table 1 comparison SHA-256 was found to be more efficient in generating the file hash as shown in Fig 12.

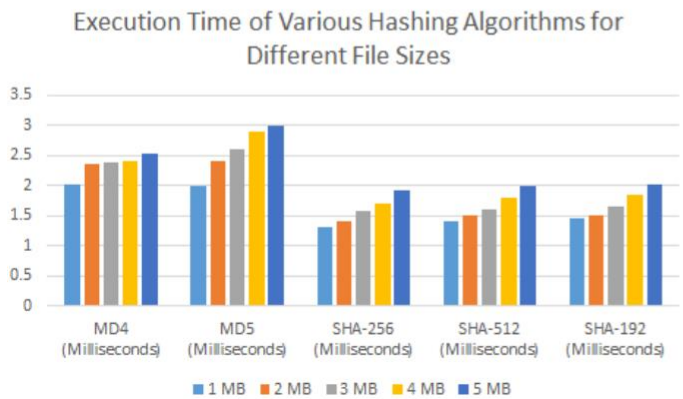


Fig 12. Performance Analysis for Different Hashing Algorithms in Generating File Hash

File Size	Symmetric Encryption Execution Time in Milliseconds			Asymmetric Encryption Execution Time in Milliseconds	
	AES	DES	Triple DES	RSA	Deffie-Hellmen
1 MB	1.32	1.58	1.89	1.50	1.59
2 MB	1.41	1.69	1.97	1.65	1.67
3 MB	1.52	1.90	2.20	1.89	1.90
4 MB	1.67	2.05	2.31	1.90	2.00
5 MB	1.89	2.03	2.42	2.00	2.32

Table 2 Comparison of Various Encryption Algorithms for Different File Sizes

From Table 2 comparison AES was found to be more efficient in encryption from symmetric cryptography as shown in Fig 13.

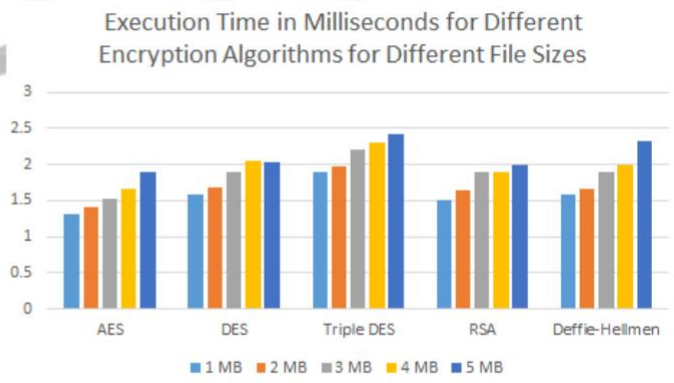


Fig 13. Performance Analysis for Different Hashing Algorithms in Generating File Hash

File Size	Huffman		7 Zip		G Zip	
	Time (Mil)	Size (KB)	Time (Mil)	Size (KB)	Time (Mil)	Size (KB)
1 MB	1.16	1.26	1.02	1.12	1.17	1.21
2 MB	1.18	1.39	1.08	1.33	1.25	1.40
3 MB	2.00	2.65	1.16	2.59	1.29	2.69
4 MB	2.05	3.26	1.95	3.19	2.32	3.42
5 MB	2.31	4.02	2.23	3.75	2.39	3.96

Table 3 Time to Compress Different File Size with Different Compression Algorithms

From Table 3 comparison 7 Zip was found to be more efficient in terms of compression and execution time as shown in Fig 14 and Fig 15 respectively.

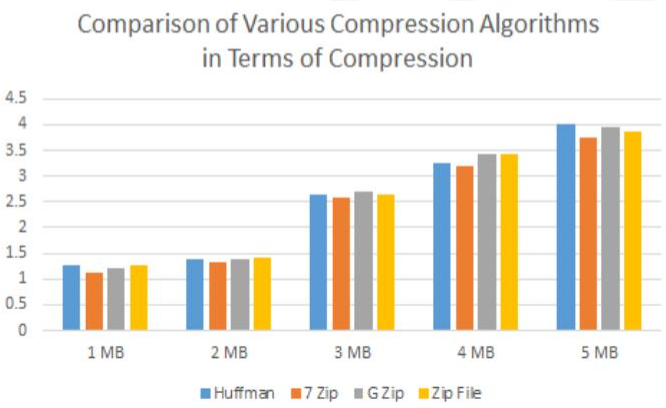


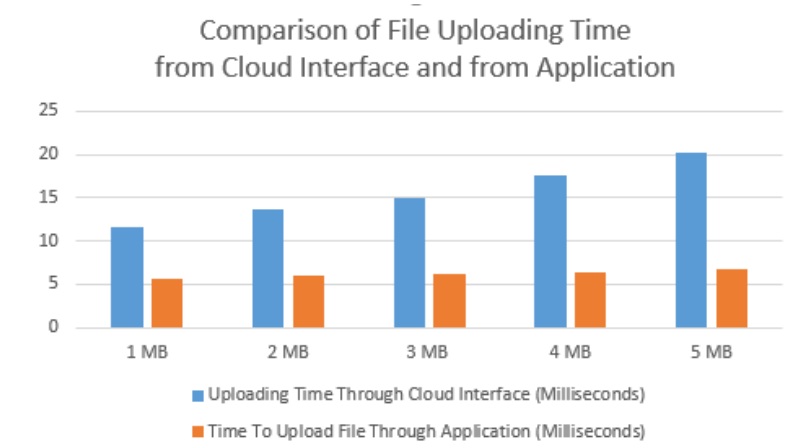
Fig 14 Performance Analysis of Different Compression Algorithms in Terms of Compression for Different File Sizes.

Fig 15 Performance Analysis of Different Compression Algorithms in Terms of Execution Time for Different File Sizes

Original File			Compressed File Package		
File Size	Upload Time (Mil)	Download Time (Mil)	File Size	Upload Time (Mil)	Download Time (Mil)
1 MB	2.32	2.20	1.32 KB	1.10	1.01
2 MB	2.39	2.31	1.53 KB	1.22	1.12
3 MB	2.41	2.32	2.79 KB	1.25	1.15
4 MB	3.45	3.30	3.39 KB	1.28	1.18
5 MB	3.53	3.12	3.96 KB	1.31	1.22

Table 4 Comparison of Uploading and Downloading Time for Original File and Compressed Package

Table 4 shows the comparison between uploading and downloading time of the original and compressed packaged file. From Fig 16 it can be easily seen.



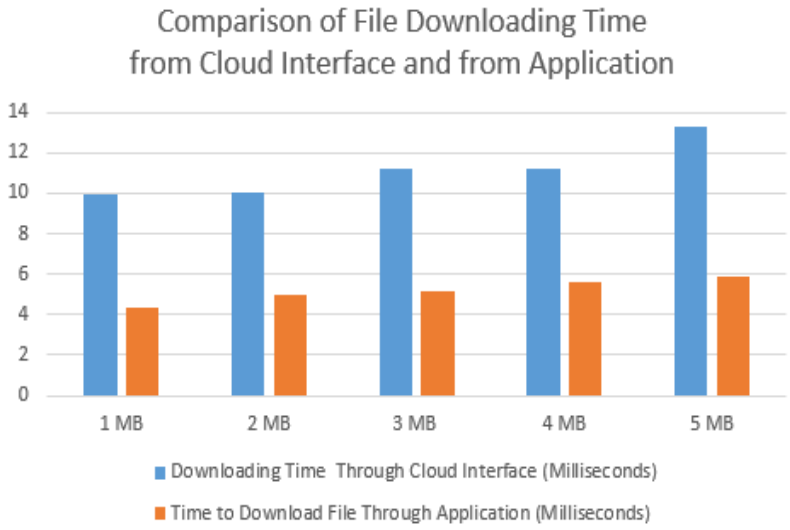


Fig 16 Performance Analysis Uploading and Downloading Time of Original and Compressed Encrypted Package

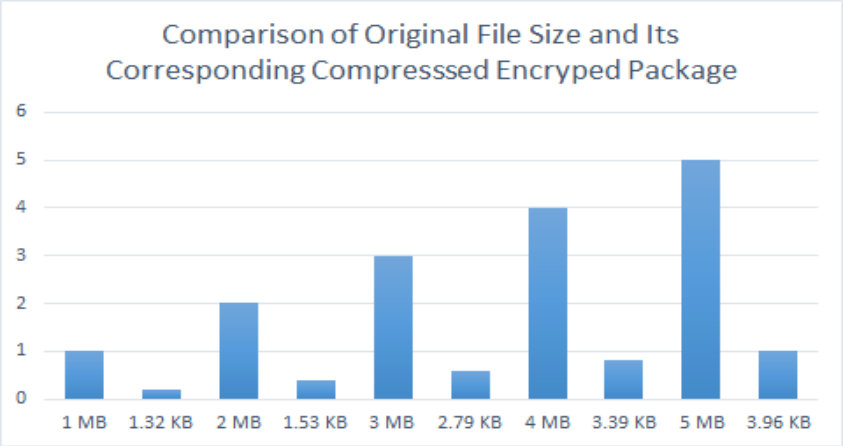


Fig. 17 Comparison of the Original File Size with its Corresponding Compressed Encrypted Package Size

File Size	Compressed Encrypted Package Size
1 MB	1.32 KB
2 MB	1.53 KB
3 MB	2.79 KB
4 MB	3.39 KB
5 MB	3.96 KB

Table 5 Comparison of Original and Compressed Encrypted Package Size

From Table 5 it can be clearly seen that proposed methodology not only improves the confidentiality and integrity of data but also reduce the storage cost. Fig 17 clearly shows the fact.

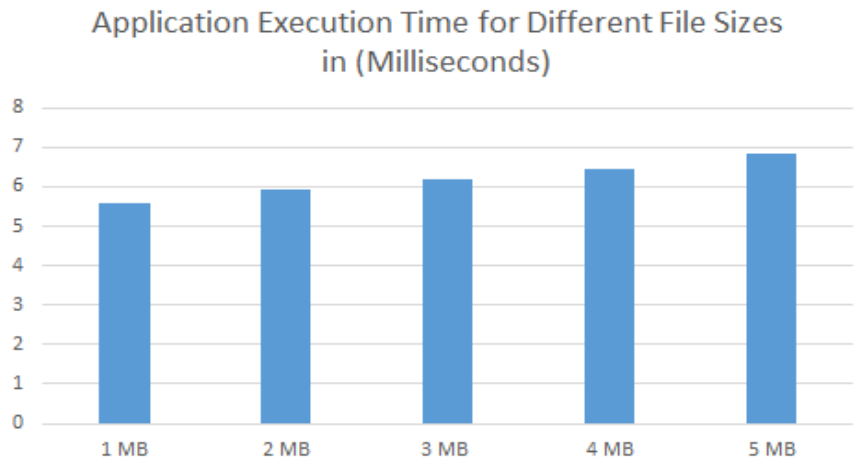


Fig. 18 Performance Analysis Application Running Time for Different File Sizes

File Size	Time to Compress (Milliseconds)			Time To Encrypt (Milliseconds)			Time To Generate Hash (Milliseconds)			Time to Upload Package (Millis)	Total Execution Time (Millis)
	7 Zip			AES			SHA-256				
	Data File	Meta File	Log File	Data File	Meta File	Log File	Data File	Meta File	Log File		
1 MB	1.21	0.10	0.12	1.25	0.13	0.13	1.22	0.15	0.16	1.10	5.57
2 MB	1.25	0.15	0.19	1.26	0.15	0.16	1.24	0.12	0.19	1.22	5.93
3 MB	1.27	0.17	0.20	1.28	0.18	0.19	1.28	0.15	0.21	1.25	6.18
4 MB	1.31	0.19	0.22	1.30	0.19	0.21	1.32	0.18	0.25	1.28	6.45
5 MB	1.41	0.22	0.25	1.32	0.22	0.25	1.35	0.22	0.28	1.31	6.83

Table 6 Application Execution Time for Different File Sizes

Table 6 is the summary of application execution time for different file sizes if n is file size in bytes then the application execution time is estimated to be T(n) ≈ (log (n)). Fig 18 is the analysis of application execution time for different file sizes.

B. **DISCUSSIONS**

The proposed methodology not only aims to provide the confidentiality and integrity to the data but also aims to provide an auditing mechanism. The purpose of this research is not only to propose a methodology not also aims to enhance the performance of the approach by using efficient sub-components (Hashing algorithm, cryptographic algorithm and compressor).

Table 1 is concerned with finding the efficient hashing algorithm to generate the hash of the file from comparison SHA-256 was found to be most suitable.

Table 2 is concerned with cryptography, different symmetric and asymmetric cryptographic algorithms were compared AES-128 was found to be more efficient from symmetric cryptography.

Table 3 is concerned with finding the most suitable compressor in terms of compression and efficiency.7 Zip compression was found to be more suitable.

Table 4 is the comparison of uploading and downloading time of various file size with their compressed encrypted version. From analysis compressed encrypted version of original file has least uploading and downloading time for each for different file sizes.

Table 5 shows the optimization of the proposed methodology in terms of space although six files are generated from a single one but the collective size of six files is even less than 10% of the original file size.

Table 6 summarizes execution time of proposed methodology for different files which is $T(n) \approx (\log (n))$ where n is file size in bytes.

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

In this research a methodology is presented to enhance the confidentiality and integrity of the data at application level. Furthermore, an auditing mechanism is also presented to avoid third party solutions. The efficiency of the proposed approach is also focused. The proposed methodology will enhance the file confidentiality and integrity by handling these aspects at application level and not relying on the third party provided solutions.

B. FUTURE WORK

In future work same approach can be applied to the multimedia objects (pictures, graphics, video and audio…etc.).

V. REFERENCES

[1]. N. Samreen and N. Khatri, "Introduction to Cloud Computing", in IRJET, Vol.5, pp. 2395-2405, 2019.

[2]. V. Suresh and M. Kumar, "An Efficient and Secure Data Storage Operations in Mobile Cloud Computing", in IJSRSET, Vol. 4, pp. 1385-1390, 2018.

[3]. A. Venkatesh, M. Eastaff, "A Study of Data Storage Security Issues in Cloud Computing", in IJSRCSEIT, Vol. 3, pp. 1741-1745, 2018.

[4]. N.Chong, "Cloud Computing Challenges in a General Perspective", in JCMS, Vol. 3, pp. 06-15, 2019.

[5]. D. Hyseni and A. Luma, "The Proposed Model to Increase Security of Sensitive Data in Cloud Computing', in IJACSA, Vol. 9, pp. 203-210, in 2018.

[6]. Vijayalakshmi and N. Veeraragavan, "A Unified Model for Cloud Data Confidentiality', in AJSAT, Vol. 7, pp. 23-27, in 2018.

[7]. Geethamani and Ranjani, "Preserving Privacy in Public Auditing for Data Storage Security in Cloud Computing", in IJSRCSEIT, Vol. 3, pp.1757-1762, in 2018.

[8] A Valerian and C. Nadunagyu, "Improvement to the Confidentiality of Cloud Data", in IJRSC,Vol 3, pp. 156-169,2018.

[9]. T. Kulkarni andV. Waghmare," Security Implementation in Cloud Computing using Behavior Profiling and Decoy Technology" in WJTER, Vol. 3, pp. 108-113, 2018.

[10] E. Agrawal and P. Ram, "Cryptography Based Security for Cloud Computing System", in IJARC,Vol. 8, pp. 2193-2197, 2017.

[11]. S. Ksasy and E. Takieldeen, "Advanced Cryptographic Algorithm System for Binary Codes by Means ofMathematical Equation", in ICIC International, Vol.10, pp. 1-8, 2016.

[12]. M. Naik and P. Tungare, "Color Cryptography using Substitution method", in IRJET, Vol. 3. pp. 941-944, in 2016.

[13] P. Sonia and S. Grewal, "Hashing Key Based Analysis of Polynomial Encryption Standard", in IJCNIS, Vol. 11, pp. 44-51, 2016.

[14] J. Puranik and A. Giri, "Security in Data Storage in Cloud Computing", in IRJET, Vol. 3, pp.1899-1902, in 2016.

[15] P. Sonia and S. Grewal, "Hashing Key Based Analysis of Polynomial Encryption Standard", in IJCNIS, Vol. 11, pp. 44-51, 2016.

[16] J. Puranik and A. Giri, "Security in Data Storage in Cloud Computing", in IRJET, Vol. 3, pp.1899-1902, in 2016.

ISSN 2320-9186

VI. AUTHORS

A. Dr. Amjad Farooq: Dr. Amjad Farooq is working as Associate Professor Department of Computer Science and Engineering University of Engineering and Technologies, Lahore. He has done Phd. From UET Lahore. His area of research is Software Engineering, Semantic Web, Bio Informatics, Image processing and Cloud computing. He has published more than 50 research papers.

B. Abdul Rehman: is student in UET Lahore Department of Computer Science. He is working as Software Engineer in InvoCode. Inc.