SSDS-MC: Slice-based Secure Data Storage in Multi-Cloud Environment

Peng Xu[1, 2], Xiaqi Liu[1, 2], Zhenguo Sheng[3], Xuan Shan[1], Kai Shuang[1]

1. State Key Lab. Of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
2. Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory, Shijiazhuang, China;
3. University of Sussex, UK.

xupeng@bupt.edu.cn, liu_xiaqi@foxmail.com, z.sheng@sussex.ac.uk, shanxuanbupt@gmail.com, shuangk@bupt.edu.cn

Abstract—**With the easy access to cloud technologies and rich applications, more people tend to store their data in cloud. However, the data in the cloud have to face the privacy and security challenges. To enforce the privacy and security protection of data in the cloud, Slice-based Secure Data Storage in Multi-Cloud Environment (SSDS-MC) is proposed in this paper. Specifically, user data are divided into multiple slices, which are stored separately into different cloud. The advantages of SSDS-MC are two-fold: i) each cloud service provider does not have all the slices of the user data and thus cannot retrieve the full user data by the slice stored in his cloud, which can further protect the privacy and security of user data; ii) each slice can be duplicated and uploaded to multiple cloud, which improve the availability of data in case the slice gets lost or damaged in one place.**

Keywords- cloud; data slice; privacy; security

I. Introduction

Cloud storage is a crucial part of cloud computing system for data storage and management [1], which provides APIs for easy access of the data [2]. Its service can access to external users through the cluster application, network technology and distributed file system, etc. In essence, user can connect to the cloud through the network at any time, from any place, and using any device [3] [4].

In the traditional storage service mode, an enterprise needs to buy servers, network communication equipments, storages equipments and human resources to build and maintain the data center, especially in the fields of IoT (Internet of Things) [5]. In contrast, cloud storage gives users more scalability and convenience on data storage [6]. Users only need to pay fees in accordance with the needs of the storage capacity, rather than to understand how to create and operate storage services. Nowadays, there are many public cloud storage services provided by famous internet companies, such as the Amazon Cloud Drive [7], Apple iCloud [8], Box [9], Microsoft SkyDrive [10] and Baidu cloud disk, etc.

However, in order to obtain a cloud storage service, the user data must be transmitted over the cloud environment. In a distributed environment of cloud storage, the user has little knowledge of where the data is stored, which makes data out of control from users. Thus, users have to rely on the cloud providers for their data privacy and security. Twinstrata's survey in 2012 showed that only 20% of people are willing to put their private data in cloud storage [11].Cloud providers are facing the trust crisis, which limits the development of cloud applications [12]. Amazon recommends users protect data security in the cloud storage service by adopting additional encryption, authentication and other security technologies [13].

In order to guarantee the privacy and security of user data, while facing the problems mentioned above, we propose the Slice-based Secure Data Storage in Multi-Cloud Environment (SSDS-MC). This paper is organized as follows. Section Ⅱ introduces related work about secure cloud storage. In Section Ⅲ, the architecture and procedures of SSDS-MC are specified. Then some key issues such as file slicing, data distribution are introduced in section Ⅳ. The experiment and the conclusions are given in the end.

II. RELATED WORDS

Security is an important criterion to measure the quality of cloud storage service. Many researchers have done a lot of work to solve the above problems.

Tang introduced the key management server in FADE system, which encrypts data twice to settle assured delete based on associated strategies and aging and provides policy-based file assured deletion with a minimal trade-off of performance overhead [14].

Shraer proposed a core set trust system in Venus. Venus is a service for securing user interaction with untrusted cloud storage to guarantee integrity and consistency for applications accessing a key-based object store service [15].

Bessani proposed the cloud in cloud idea in DEPSKY, DEPSKY improves the availability, integrity and confidentiality of information stored in the cloud through the encryption, encoding and replication of the data on diverse clouds that form a cloud-of-clouds, which mitigate the impact of the problem on data confidentiality and vendor data lock-in to a certain extent [16].

As can be seen from the existing research results, data encryption and access control mechanisms are widely used to protect the security and privacy of user data in the cloud storage. Data encryption is able to prevent unauthorized dissemination, but the system needs to provide other complex mechanisms to manage encryption keys and these management mechanisms

QSHINE 2015, August 19-20, Taipei, Taiwan
Copyright © 2015 ICST
DOI 10.4108/eai.19-8-2015.2260679

might lead to other usability problems [17]. For example, in FADE, if the key management server is destroyed or tampered, all data is no longer accessible. DEPSKY also requires a secret sharing scheme. Access control mechanisms can prevent a user without the permission to access the data in proper channels, but because the access control component is running in untrustworthy cloud environment, that may not guarantee the correct implementation of user-defined access control policy. For that reason, Venus has to provide security functionality to users through extra tripartite framework.

III. DESIGN OF SSDS-MC

This section presents the SSDS-MC system. It starts by presenting the system architecture, and then introduces the procedures of SSDS-MC system.

Architecture of SSDS-MC is shown in figure 1. It is composed of four main components, which are Data Processing (DP), Data Distribution (DD), Data Collection (DC) and Metadata Table (MT).

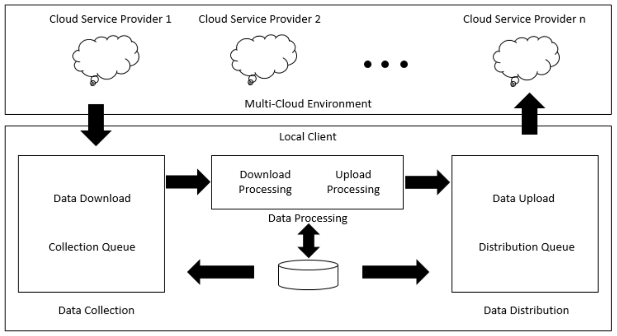


Figure 1. Architecture of SSDS-MC

- **Data Processing (DP):** This module as shown in figure 2 has two functions: when uploading files, this module is responsible for encrypting the file using Data Encryption Algorithm and slicing files before uploading them to multi-clouds following Data Slicing Policy; when downloading files, this module is responsible for recovering encrypted files from slices downloaded from multi-clouds and decrypting them into the original files;

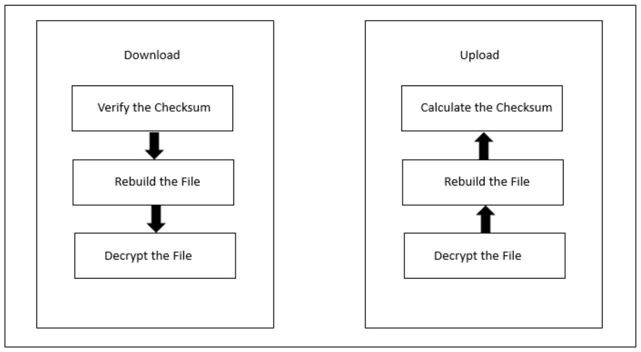


Figure 2. Procedure of Data Processing(DP)

- **Data Distribution (DD):** This module is used to upload data slices to multi-clouds following certain Data Distribution Policy;
- **Data Collection (DC):** Data Collection functions as downloading data slices from multi-clouds based on the information in Metadata Table. SSDS-MS stores multiple backups separately for each data slice to allow users to get backup data slice when the default one is unavailable. Every time, DC first downloads slices from default cloud;
- **Metadata Table (MT):** MT records the mapping information of files, their corresponding slices and slice storage information. Metadata in the MT is described in Table I.

TABLE I. METADATA IN METATABLE

Belonging File Info.	File name; …
Slice Sequence Number/Number of Slices	
Slice Checksum	
Cloud Provider Info. For Default block	Cloud Provider Interface Address; Account info.; Directory Info. …
Cloud Provider Info. For Other backups	Cloud Provider Interface Address; Account info.; Directory Info. …
Last Update Time	YY/MM/DD

The procedures of SSDS-MC system includes uploading and downloading files.

In SSDS-MC, Data uploading procedure is listed as follows:

1) *By utilizing the data encryption algorithm, DP encrypts a file and the private key is assigned by the user.*
2) *DP slices the encrypted file into multiple data slices following data slicing policy.*
3) *DP calculates the checksum of each slice, which would be used to verify the downloaded slices. The slice information and the corresponding checksum is supposed to be recorded into MT.*
4) *DP would follow data distribution policy to generate certain data distribution information for each slice of user data and the data distribution information is also recorded in the MT.*
5) *File slices are stored to the uploaded queue of DD after being sliced. Every time, DD takes some of the slices from the upload queue and follows MT to upload data slices into cloud. Because a user file is sliced into multiple slices and all these slices could be uploaded simultaneously, which effectively improves the upload speed of user data.*

Furthermore, procedure of data downloading is shown as follows:

1) DC queries default download information from MT and download all data slices from multi-clouds. File slices are saved in collection queue. Once a default server of certain slice is unavailable, DC will try to lookup backup slice information from MT and complete the download task.

2) DP collects slices from collection queue, calculates their checksum and compares them to the data in MT, if the checksum for some slices is incorrect; re-executing the downloading assignment is required.

3) DP stitches all the slices to an encrypted file and decrypting it with private key.

IV. KEY ISSUES OF SSDS-MC

As mentioned above, there are many key issues that are supposed to be solved in SSDS-MC, such as data encryption algorithm, data slicing policy and data distribution policy.

A. Data Encryption Algorithm

Data Encryption Algorithm can guarantee that others are not able to recover the user data even if they get several slices of user data.

There are two main categories of cryptography depending on the type of security keys used to encrypt/decrypt the data. These two categories are asymmetric and symmetric encryption techniques [18] [19]. Symmetric encryption is faster, while it is less secure than asymmetric encryption. In the SSDS-MS system, a large number of data required to encryption and decryption, thus we incline to select a symmetric encryption algorithm, which is faster in general [19]. However, an asymmetric encryption can be used to manage keys of symmetric encryption. In this way, we combine the advantages of Asymmetric and Symmetric encryption techniques [18] [19].

In this paper, three secret key encryption algorithms are considered.

- **DES:** (Data Encryption Standard), was the first encryption standard to be published by NIST (National Institute of Standards and Technology). DES uses a 56-bit key, and maps 64 bit input block into a 64-bit output block. The key actually looks like a 64-bit quantity, but one bit in each of the 8 octets is used for odd parity on each octet [20]. However, there are many methods can attack the weaknesses of DES, which made it an insecure block cipher [21]. The procedure of DES encryption algorithm is shown in figure 3.

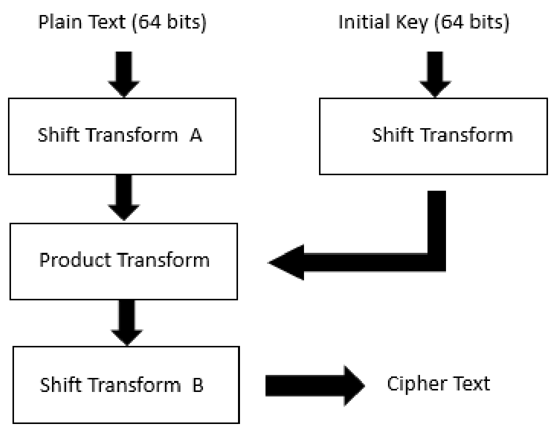


Figure 3. The Procedure of DES

- **3DES:** 3DES is a proposal based on DES, which standardized in ANSI X9.17 & ISO 8732 and in PEM for key management. It is backwards compatible with existing single DES (when K1=K2=K3). The 3DES algorithm uses either two or three 56-bit keys. Thus, the length of effective key is summed up to 168 bits [19]. 3DES is defined by the following function:

$$\mathrm{C} = DES_{K3}\{DES_{-1}$$
$$_{K2}\{DES_{K1}(P)\} \qquad (1)$$

Where P = Plaintext,

C = Ciphertext

DES-1

K = DES decryption using key K

- **AES:** (Advanced Encryption Standard), is also known as the Rijndael(pronounced as Rain Doll) algorithm, which is a symmetric block cipher that can encrypt data blocks of 128 bits using symmetric keys 128, 192, or 256. AES was introduced to replace the DES [22]. Brute force attack is the only effective attack known against this algorithm [23]. Procedure of AES is represent in Figure 4.

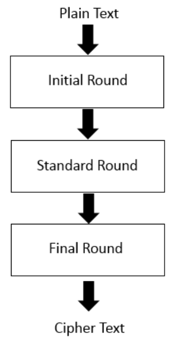


Figure 4. The Procedure of AE

TABLE II. COMPARISON OF ENCRYPTION ALGORITHM

	DES	3DES	AES
Developed time	1977	1978	2000
Block size	64bits	64bits	128,192 or 256bits
Key length	56bits	K1=K2=168bits,K3=112bits	128,192 or 256bits
Number of rounds	16	48	9,11 or 13
Security	Proven inadequate	More security than DES	Considered secure

TABLEⅡ lists the comparison of these kinds of symmetric encryption algorithm, from the key length, safety and other aspects [24].

To ensure the safety, in the prototype proposed in this paper, AES with a key of 128bits is involved.

B. *Data Slicing Policy*

In SSDS-MC, a file is sliced into many slices. To slice data into fixed-size slices might keep the implementation of SSDS-MC simple. When the slice size is too small, the file is sliced into more slices, which might consume unnecessary transmission resources for more connections with cloud providers. Once the data slices are too large, computing tasks for data slices would occupy too more memory resources. Therefore, different slice size should be applied based on the file size. Data Slicing Policy is used to decide the slice size for each user file when uploading it.

In general, the more slices, more safety the data is, but more resources consumption are required to complete the calculation task. In that case, we recommend slicing data and upload slices at the same time when the file is too large. To be specific, the system read bytes of a certain size from the file stream and distribute to data centers immediately, which can avoid PC to a read a completely large file into memory, multi threads also can be adopted to the system.

It is difficult to determine a certain value as the optimal fixed size as different environment may provide different computing capability. In the experiment mentioned in the Ⅴ section, we first slice a 100M file for different size, i.e. 100kb、1M and 10M , then observe and analysis experimental data in system performance and security to find a relatively better number as the fixed size.

C. *Data Distribution Policy*

Data Distribution Policy decides which cloud a certain data slice should be stored. There are many chances that cloud servers might lose or corrupt data leading to data unavailability. The same problem may also be caused by unavailable of cloud services. Therefore, multiple copies of each slice should be distributed to multi-cloud in SSDS-MC, which allows users to access another copy of the data slice from another cloud when the default copy is unavailable. It means that Data Distribution Policy should select multiple clouds for each data slice and set one cloud as default.

On deciding the default cloud, many factors like transform speed and fees for buying the storage service supposed to be taken into account. It can be an option that picking out the nearest cloud as default. Alternatively, select the cloud with least money expend as the default storage. It is more logical to select the default cloud service in accordance with the percentage of user expected. Data Distribution Policy should avoid all default data saved in one cloud service, which will reduce the security of the system.

V. EXPERIMENT

The prototype of SSDS-MC is implemented with python. A library called PyCrypto provide APIs for various encryption algorithm. The prototype slices data after data encryption. Codes of the prototype contain two parts: ⑴ File Reading Module and Slice Outputting Module, which can read a file as binary stream and slice it based on the fixed size we set. ⑵Encryption Module. This module encrypts binary plaintext using the AES algorithm and outputs ciphertext.

A. *Best-fixed Size*

In order to find out the best-fixed size in the experiment environment, we conduct an experiment to record time cost on slicing a file into a series size. We consider 100M and 1G as two-file size magnitude. Time cost on splitting a file into different sizes shown in tableⅢ. The experimental environment is a PC with 4G memory and 2.5 Hz.

TABLE III. TIME COST ON SPLIT WITH DIFFERENT SIZES

File Size	100kb	1M	10M
100M	1.591s	0.156s	0.14s
1G	39.13s	20.436s	21.543s

It can be seen from the results, when a file is split into too many pieces, there will be a large time-consuming for system to accomplish task, thus decreasing system availability. Nevertheless, when it is cutted into very few blocks, we did not see significantly reduce on time-consuming. For a specific computing environment, there is a relatively appropriate fixed size when considering the system performance and safety requirement together.

In the following experiment, we chose 1M as the fixed size slice.

B. *Performance on Data Processing*

The second set of experiment is conducted to find out time consumption on data processing, the results are shown in Figure 5.

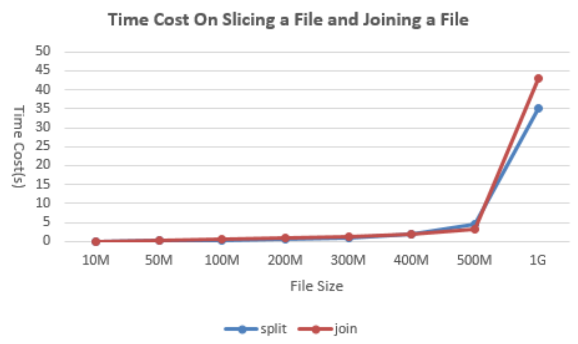


Figure 5. Time Cost On Slicing a File and Joining a File

Time-consuming for file processing increases with the size of the file. When a single file size is not more than 300M, the time consumption less than one second. While when the file size is close to 1G, time-consuming has a significantly increase (consuming as high as 43s).

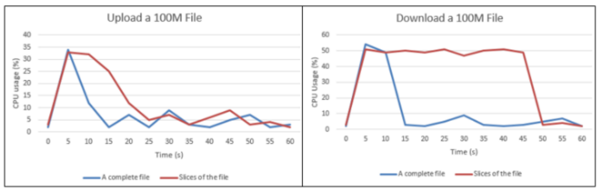
The results depicted in the figure show that time cost of data processing has little effect on system efficiency when processing small files. In such a situation, SSDS-MC can enhance the storage security by exchanging with minimal loss on time depletion. However, when the file size exceeds a certain threshold, the cost becomes unable to ignore. Therefrom, we conclude that SSDS-MC is more suitable for storing small files, which need safety insurance, such as documents and pictures, while it does not applicable for video, and other large files.

C. Performance on Uploading and Downloading

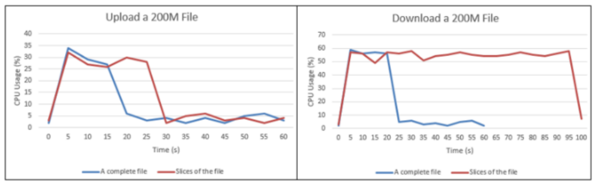
We use one PC to simulate the cloud though a real cloud server may have a stronger computing power. Another PC will behave as a user connecting to the cloud through FTP protocol to simulate a real Internet environment.

In the experiment, we upload and download files by two methods : (1). Upload and download a complete file, (2). Upload and download its slices by opening six threads for this task.

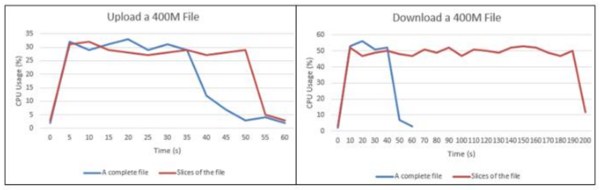
First, we measure the CPU usage of both two methods when the file size are 100M, 200M and 400M. The result in Figure 6 reveals that the maximum CPU occupies (curve peak) for two methods has a very small gap. To be more precisely, upload and download slices occupancy CPU less (from 1% to 2% percent) than a complete file. However, upload and download slices takes more time. It is no surprise to notice that because upload and download slices expend more time on establishing connections.



(a). CPU usage in upload and download a file of 100M



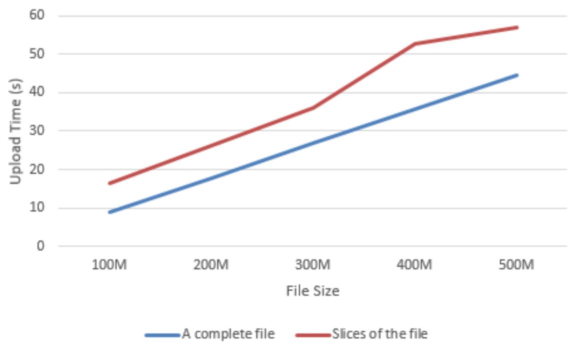
(b). CPU usage in upload and download a file of 200M



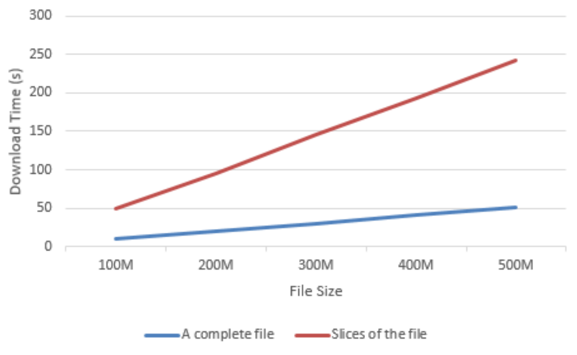
(c). CPU usage in upload and download a file of 400M

Figure 6. CPU usage in uploading and downloading

We also recorded time comparison of above two ways. Figure 7a and figure 7b correspond to the process of uploading and downloading. We can know that compared to upload and download a complete file directly, process its slices spend more time. In addition, the time gap growing with file size. This once again confirms our previous conclusions: SSDS-MC is more suitable for storing small files that need safety insurance. We strongly recommend that each slice be uploaded to the cloud or downloaded from the cloud simultaneously, which could improve the upload and download speed largely.



(a). Time comparison of a 200M file



(b). Time comparison of a 400M file

Figure 7. Time Cost on downloading and uploading

VI. CONCLUSION

In this paper, an innovative solution "Slice-based Secure Data Storage in Multi-Cloud Environment" for secure data storage based on data slicing is proposed. It is verified by the prototype that SSDS-MC can addresses several advantages for small files as follow:

- **Privacy and Security of Data:** SSDS-MC deals with this problem by data slicing and storing slices into multi-clouds. Meanwhile, SSDS-MC encrypts files before slicing them. Therefore, each cloud service provider could not recovery user data with incomplete and encrypted user data.
- **Availability of Data:** SSDS-MS stores multiple backups separately for each data slice to allow user get backup data slice when the default one is unavailable.

We can get a conclusion by experiment that SSDS-MC is suit for storing small files that require security and privacy, because it can help protect data, meanwhile the data pre-process and uploading or downloading process do not have noticeable influence on system efficiency. For large files, SSDS-MS can still provide a solution if the data security requirements higher than the efficiency of the system.

In the future, we will further study how to combine SSMS-DS with a feasible Data Distribution Policy.

ACKNOWLEDGMENT

This work was funded by the open project of Science and Technology on Information Transmission and Dissemination in Communication Networks Laboratory (ITD-U14003/KX142600010).

REFERENCES

[1] ShiQiang,ZhaoPengyuan. Analysis of critical technologies on cloud storage security. Journal of the Hebei Academy of Sciences, Vol.28 No.3, Sep.2011.

[2] Armbrust, M et al. Above the Clouds: A Berkley View of Cloud Computing. UC Berkley Technical Report, 2009.

[3] C. Wang, Q. Wang, K. Ren, and W. Lou: Ensuring data storage security in Cloud Computing. IWQoS 2009: 1-9

[4] M. Chen, L. Hu, Y. Zhang, T. Taleb and Z. Sheng, "Cloud-based Wireless Network: Virtualized, Reconfigurable, Smart Wireless Network to Enable 5G Technologies", ACM/Springer Mobile Networks and Applications (MONET) Special Issue on Networking 5G Mobile Communications Systems, 2015.

[5] Z. Sheng, S. Yang, Y. Yu, A. V. Vasilakos, J. Mccann, and K. K. Leung "A Survey on The IETF Protocol Suite for The Internet-of-Things: Standards, Challenges and Opportunities", IEEE Wireless Communication Magazine, vol.20, no.6, pp.91,98, December 2013.

[6] B. Butler. Personal Cloud Subscriptions Expected to Reach Half a Billion This Year. In Network World, September 7 2012

[7] Amazon Simple Storage Service (S3): aws.amazon.com/s31

[8] iCloud. https://www.icloud.com/.

[9] Box. https://www.box.com.

[10] Microsoft OneDrive. https://onedrive.live.com/.

[11] Twinstrata,http://www.twinstrata.Com, Oct. 05 2012.

[12] Network security lab of china telecommunications.Cloud computing security - technology and applications.Beijing:Publishing house of electronics industry,2012.

[13] S. L. Garfinkel. An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS. In Tech Report TR-08-07, Harvard University, 2008.

[14] Y. Tang, P. Patrick.C. Lee, John C.S. Lui, Radia Perlman, "FADE : Secure overlay cloud storage with file assured deletion," Proc. The 6th Int Conf on Security and Privacy in Communication Networks, pp. 380–397, 2010.

[15] S. Alexander, C. Christian, C. Asaf, K. Idit, M. Yan, S. Dani, "Venus: verification for untrusted cloud storage," Proc. The 2010 ACM Workshop on Cloud Computing Security Workshop, pp. 19–30, 2010.

[16] B. Alysson, C. Miguel, Q. Bruno, A. Fernando, S. Paulo,"DEPSKY: dependable and secure storage in a Cloud-of-Clouds,"Proc. The 6th Conf on Computer System, pp. 31–46, 2011.

[17] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, Jin Li, "Toward Publicly Auditable Secure Cloud Data Storage Services ‖ , IEEE Network, 2010.

[18] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption—How to encrypt with RSA" Advances in Cryptology-EUROCRYPT'94.

[19] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation", Proceedings of FOCS97, IEEE, 1997.

[20] National Bureau of Standards - Data Encryption Standard, FIPS Publication 46, 1977.

[21] Singh, S Preet and Maini, Raman. "Comparison of Data Encryption Algorithms", International Journal of Computer Science and Communication, vol. 2, No. 1, January-June 2011, pp. 125-127.

[22] NIST, "Advanced Encryption Standard Call", NIST, 1997. http://www.nist.gov/aes/

[23] NIST Advanced Encryption

[24] Singhal, Nidhi and Raina, J P S. "Comparative Analysis of AES and RC4 Algorithms for Better Utilization", International Journal of Computer Trends and Technology, ISSN: 2231-280, July to Aug Issue 2011, pp. 177-181.