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. 2, Issue 11, November 2014

Enhancing Data Securing In Cloud Using Scalable Transactions

Mrudul S Rajhans

Department of Computer Science, PICT, University of Pune, MH, India

ABSTRACT: Cloud computing is becoming one of the most used paradigms to deploy highly available and scalable systems. These systems usually demand the management of huge amounts of data, which cannot be solved with traditional nor replicated database systems as we know them. Recent solutions store data in special key-value structures, in an approach that commonly lacks the consistency provided by transactional guarantees, as it is traded for high scalability and availability. In order to ensure consistent access to the information, the use of transactions is required. However, it is well-known that traditional replication protocols do not scale well for a cloud environment. Here we take a look at current proposals to deploy transactional systems in the cloud and we propose a new system aiming at being a step forward in achieving this goal. We proceed to focus on data partitioning and describe the key role it plays in achieving high scalability. No SQL Cloud data stores provide scalability and high availability properties for web applications, but at the same time they sacrifice data consistency. However, many applications cannot afford any data inconsistency. Cloud TPS is a scalable transaction manager which guarantees full ACID properties for multi-item transactions issued by Web applications, even in the presence of server failures and network partitions. We implement this approach on top of the two main families of scalable data layers: in our local cluster and Amazon Simple DB inthe Amazon cloud shows that our system scales linearly at least up to some nodes in the Amazon cloud. The given cloud implementation has done on open stack framework using Ubuntu operating environment.

KEYWORDS: Data security, cloud computing, Cryptography techniques, Distributed system, AES(Advance Encryption Standard).

I. INTRODUCTION

The use of multiple database instances to maintain security instead of single database instance. This work mainly focus on sharing document on cloud using AES(Advanced Encryption Standard)[1] encryption algorithm. It actually divide single document in to two instances. Where these instances are reside on single cloud. It also support scalable transaction of document.

Motivation of Dissertation

Cloud computing is becoming wider and popular and thus providing a vibrant technical environment where innovative solutions and services can be created. Cloud promises its end users cheap and flexible services.It offers the vision of a virtually infinite pool of computing, storage and networking resources where applications can be scalable deployed[4]. Cloud computing provides its users with different types of services. One of the most important service provided by cloud is STaaS[16] i.e. storage as a service.Customers often store sensitive information with cloud storage providers. Thus providing a secure framework in the cloud computing environment is the challenge which is being faced by the cloud storage providers. Customers want their data to be secured as well as it should be available at any time. Many a times this becomes difficult with a single cloud provider and it may result in service availability failure as well as the possibility of data intrusion or data getting stelling from the cloud provider. To overcome these failures we are using AES(Advanced Encryption Standard) algorithm for encryption [1]and data will be divided into two or more database servers so single whole data is not available at single server.

The remainder of this paper is organized as follows.
Section 2 describes Literature survey of paper.It gives references of paper and what they suggest. Section 3 describe

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. 2, Issue 11, November 2014

Implementation detail of our system with mathematical model , system architecture and algorithms used.Section 4will concludethe paper.

II. LITERATURE SURVEY

Abha, Mohit propose a simple data protection model where data is encrypted using Advanced Encryption Standard (AES) before it is launched in the cloud, thus ensuring data confidentiality and security.

Mr.Mohammedet al. [1], found that the research into the use ofmulti-cloud providers to maintain security has received less attention from the research community than has the use of single clouds. This work aims to promote the use of multi-clouds due to its ability to reduce security risks that affect the cloud computing user.[3]

Mr.CongWang et al. [5] investigates secure outsourcing of widely applicable linear programming (LP) computations. In order to achievepractical efficiency, this mechanism design explicitly decomposesthe LP computation outsourcing into public LP solvers runningon the cloud and private LP parameters owned by the customer.

Zhou Wei, Guillaume Pierre, Chi-Hung Chi et sl[4] search thatCloudTPS is a scalable transaction manager which guarantees full ACID properties for multi-item transactions issued by web applications, even in the presence of server failures and network partitions. So they implement this approach on top of the two main families of scalable data layers: Bigtable and SimpleDB. Performance evaluation on top of HBase (an open-source version of Bigtable) in our local cluster and Amazon SimpleDB in the Amazon cloud shows that our system scales linearly at least up to 40 nodes in our local cluster and 80 nodes in the Amazon cloud.

K.D. Bowers, A. Juels and A. Opreaintroduce HAIL (High-Availability and Integrity Layer), a distributedcryptographic system that allows a set of servers to prove toa client that a stored file is intact and retrievable.

DebajyotiGitesh, Parthi, Sagar ,Vibha suggest the encryption of the files to be uploaded on the cloud. The integrity and confidentiality of the data uploaded by the user is ensured doubly by not only encrypting it but also providing access to the data only on successful authentication.

III. IMPLEMENTATION DETAILS

Mathematical model of System

Let S be the system that use cloud for storing documents created and used by different users. In this cloud use two instance for dividing document in encrypted format where input by user is plaintext but output is in cipher text.Data will be stored in linux image's instance.Document must be divided in scalable manner means if large document come it require appropriate time.

$S= \{I, O, F, Su, Fa / \phi_S\}$

Where

- I is the input to the system.
- O is the output of the system.
- F is set of functions.
- Su is success of system.
- Fa is the failure of the system.

- INPUT:

o I is the input set such that

– $I = \{U,D,V\}$.

– $P = \{u_1,u_2,u_3\ldots.u_n\}$, set of 'n' users.

– $V = \{v_1, v_2, v_3\ldots.v_m\}$, set of 'm' virtual machines.

– $D = \{ d_1,d_2,d_3\ldots.d_m \}$, set of documents used.

- OUTPUT:

o Output (O) is the cipher text data will be stored in database.

– Output (O) = D

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. 2, Issue 11, November 2014

- FUNCTIONS
  - o F is a set of functions where.

– F = {F_p, F_c}

  - o F_p is a function for entering plaintext by user.
  - o F_c is a function for entering ciphertext by user.
- SUCCESS

Large document and small document require adequate time

- FAILURE
  - o Document is not successfully divided into two instances.

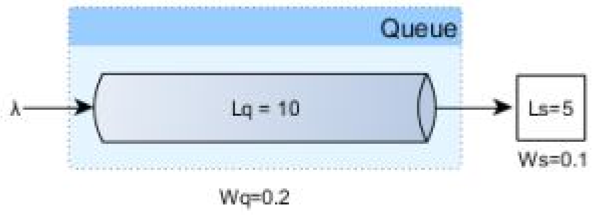
Scalable Transactions in our approach

We operates with long-term averages and it might not suit for various traffic burst patterns. That's why metrics are very important when doing resource provisioning. The queue is valuable because it buys us more time. It doesn't affect the throughput. The throughput is only sensible to performance improvements or more servers. But if the throughput is constant then queuing is going to level traffic bursts at the cost of delaying the over flown requests processing.

Ws = service time (the connection acquire and hold time) = 100 ms = 0.1s
Ls = in-service requests (pool size) = 5
Assuming there is no queueing (Wq = 0):

$$\lambda = \frac{L}{W} = 50\frac{requests}{s}$$

Our connection pool can deliver up to 50 requests per second without ever queueing any incoming connection request. Whenever there are traffic spikes we need to rely on a queue, and since we impose a fixed connection acquire timeout the queue length will be limited.



Since the system is considered stable the arrival rate applies both to the queue entry as for the actual services:

$$\lambda = \frac{Ls}{Ws} = \frac{5}{0.1} = \frac{Lq}{Wq} = \frac{10}{0.2}$$

This queuing configuration still delivers 50 requests per second but it may queue 100 requests for 2 seconds as well.
A one second traffic burst of 150 requests would be handled, since:

- 50 requests can be served in the first second
- the other 100 are going to be queued and served in the next two seconds

The timeout equation is:

$$Lspike = \lambda spikeTspike$$
$$T = \frac{Lspike}{\lambda} = \frac{\lambda spikeTspike}{\lambda}$$
$$Lq = Lspike - Ls$$
$$Tq = T - 1$$

So for a 3 seconds spike of 250 requests per second:
Λ spik250requests/s
Tspike = 3s
The number of requests to be served is:

$$Lspike = 250\frac{requests}{s}3s = 750requests$$
$$T = \frac{750requests}{50\frac{requests}{s}} = 15s$$

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. 2, Issue 11, November 2014

$$Lq = Lspike - Ls = 700requests$$
$$Tq = T - 1 = 14s$$

This spike would require 15 seconds to be fully processed, meaning a 700 queue buffer that takes another 14 seconds to be processed.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system aims to build private cloud using open source software OpenStack. The system architecture ofOpenStackis as depicted in Fig.1. The proposed system consists of various modules such as Horizon, Nova, Swift, Glance, and Keystone

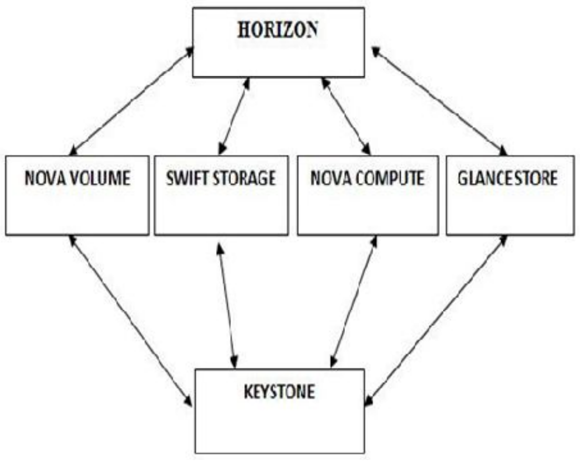


Figure 1: OpenStack System Architecture

- ***Nova:*** Nova is the Computing Fabric controller for the OpenStack Cloud. The necessary activities for the life cycle of instances within the OpenStack cloud are handled by Nova. This characteristic makes Nova a Management Platform to manage various compute resources, networking, authorization, and scalability needs of the OpenStack cloud.
- ***Glance:*** Glance is a standalone service which provides a catalog service for storing and querying virtual disk images. Nova and Glance together provides an end-to end solution for cloud disk image management.
- ***Swift:*** Swift can store billions of virtual object distributed across the nodes. The swift offers built-in redundancy, failover management, archiving and media streaming. Swift plays an important role in scalability.
- ***Keystone:*** Keystone provides identity and access policy services for all components in the OpenStack family.All components of OpenStack including Swift, Glance, and Nova are authenticated and authorized by Keystone.
- ***Horizon:*** Horizon can be used to manage instances and images, create key pairs, attach volumes to instances,manipulate Swift containers etc.

V. IMPLEMENTATION

The proposed system is implemented using open source software called Openstack and Ubuntu operating system.The three nodes such as Compute, Controller and Storage are installed with Ubuntu server operating system becauseall these nodes have to behave like servers as shown in figure Compute node is installed with the Nova packagesand services. Controller node is installed with the Glance,Keystone and Horizon packages and services. Storage node is installed with the Swift or cinder packages andthe services. All three nodes are connected internally to OpenStack Dashboard with internal network. TheApplication which is ready to use the cloud service is connected through external network to controller node of
the private cloud.

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. 2, Issue 11, November 2014

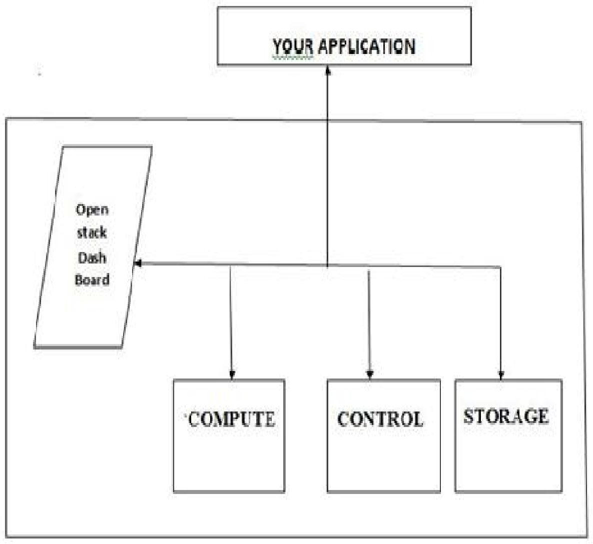


Figure. 2: Open Stack Implementation Architecture

Module 1: Compute Node

The installation of nova packages is carried out by downloading the nova packages by the following Command:

- sudo apt-get install nova-api nova-cert nova-compute nova-compute-kvm nova-doc nova-network nova object store nova-scheduler nova-volume rabbitmqservernovnc nova-consoleauth These install lines added most of the packages that expected (nova-api, nova-compute, nova-network etc.) to work nova on the open stack.

Module 2: Control Node

The installation of glance packages is carried out by downloading the glance packages by the following command sudo apt-get install glance glance-api glance-client glance-common glance-registry python- glance These install lines added most of the packages that expected (glance-api, nova-registry etc.) to work on the open stack. The installation of keystone packages is carried out by downloading the keystone packages by the following command.

- sudo apt-get install keystone python-keystone python key stone client These install lines added most of the packages that expected (python-keystone, python-keystone etc.) to work keystone on the open stack. The installation of horizon packages is carried out by downloading the horizon packages by the following command.
- sudo apt-get install open stack-dashboard These install lines added most of the packages that expected to work dashboard on the OpenStack.

Module 3: Storage Node

downloading the swift packages by the following command

- sudo apt-get install swift swift-proxy swift-account swift-container swift-object These install lines added most of the packages that expected (swift-proxy, swift-account, swift-container etc.) to work swift on the OpenStack.

Algorithms

AES

The plaintext input and cipher text output for the AES(Advanced Encryption Standard) algorithms are blocks of 128 bits. The cipher key input is a sequence of 128, 192 or 256 bits. In other words the length of the cipher key, N_k, is either 4, 6 or 8 words which represent the number of columns in the cipher key. The AES(Advanced Encryption Standard) algorithm is categorized into three versions based on the cipher key length. The number of rounds of encryption for each AES(Advanced Encryption Standard) version depends on the cipher key size.

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. 2, Issue 11, November 2014

In the AES(Advanced Encryption Standard) algorithm, the number of rounds is represented by N_r, where $N_r = 10$ when $N_k= 4$, $N_r= 12$ when $N_k= 6$, and $N_r= 14$ when $N_k= 8$. The following table illustrated the variations of the AES(Advanced Encryption Standard) algorithm. For the AES(Advanced Encryption Standard) algorithm the block size (N_b), which represents the number of columns comprising the *State* is $N_b= 4$.

Table 1 – AES(Advanced Encryption Standard) Variations

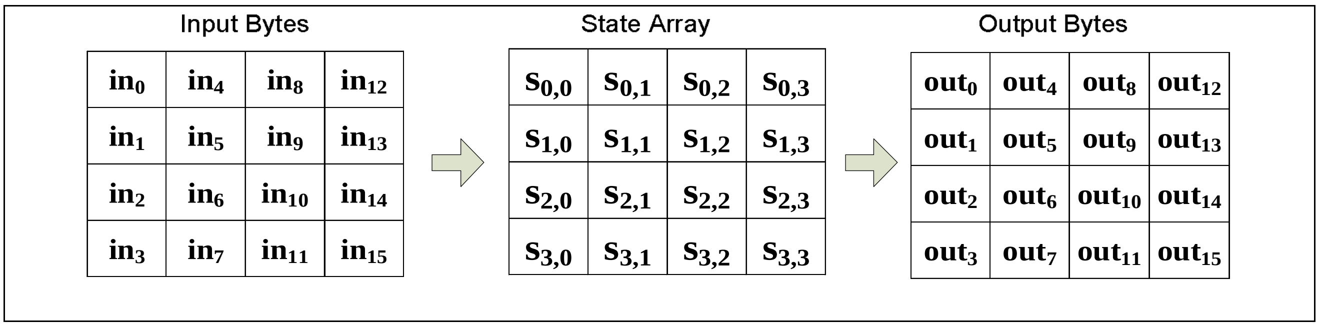
AES Version	Key Length (N_k words)	Block Size (N_b words)	Number of Rounds (N_r rounds)
AES128	4	4	10
AES192	6	4	12
AES256	8	4	14

The basic processing unit for the AES(Advanced Encryption Standard) algorithm is a byte. As a result, the plaintext, cipher text and the cipher key are arranged and processed as arrays of bytes. For an input, an output or a cipher key denoted by *a*, the bytes in the resulting array are referenced as a_n , where *n* is in one of the following ranges:
Block length = 128 bits, 0 <= *n* < 16
Key length = 128 bits, 0 <= *n* < 16
Key length = 192 bits, 0 <= *n* < 24
Key length = 256 bits, 0 <= *n* < 24
All byte values in the AES(Advanced Encryption Standard) algorithm are presented as the concatenation of their individual bit values between braces in the order {*b*7, *b*6, *b*5, *b*4, *b*3, *b*2, *b*1, *b*0}. These bytes are interpreted as finite field elements using a polynomial representation:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x + b_1x + b_0x = \sum_{i=0}^{7} b_i x^i$$

As an example, {10001001} (or {85} in hexadecimal) identifies the polynomial $x^7 + x^3 + 1$. The arrays of bytes in the AES(Advanced Encryption Standard) algorithm are represented as $a_0a_1a_2...a_n$.

All the AES(Advanced Encryption Standard) algorithm operations are performed on a two dimensional 4x4 array of bytes which is called the *State*, and any individual byte within the *State* is referred to as $s_{r,c}$, where letter '*r*' represent the row and letter '*c*' denotes the column. At the beginning of the encryption process, the *State* is populated with the plaintext. Then the cipher performs a set of substitutions and permutations on the *State*. After the cipher operations are conducted on the *State*, the final value of the state is copied to the ciphertext output as is shown in the following figure.



in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Fig 3 – State Population and Results

At the beginning of the cipher, the input array is copied into the *State* according the following scheme:
s[r,c] = in [r + 4c] for $0 \le r < 4$ and $0 \le c < 4$,

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. 2, Issue 11, November 2014

and at the end of the cipher the *State* is copied into the output array as shown below:
out[r+4c] = s[r,c] for $0 \le r < 4$ and $0 \le c < 4$

Cipher Transformations

The AES(Advanced Encryption Standard) cipher either operates on individual bytes of the *State* or an entire row/column. At the start of the cipher, the input is copied into the *State* as described in Section 2.2. Then, an initial *Round Key* addition is performed on the *State*. Round keys are derived from the cipher key using the *Key Expansion* routine. The key expansion routine generates a series of round keys for each round of transformations that are performed on the *State*.

The transformations performed on the state are similar among all AES(Advanced Encryption Standard) versions but the number of transformation rounds depends on the cipher key length. The final round in all AES(Advanced Encryption Standard) versions differs slightly from the first N_r□1 rounds as it has one less transformation performed on the *State*. Each round of AES(Advanced Encryption Standard) cipher (except the last one) consists of all the following transformation:

- SubBytes( )
- ShiftRows( )
- MixColumns( )
- AddRoundKey ( )

The AES(Advanced Encryption Standard) cipher is described as a pseudo code in Figure 3. [1] As shown in the pseudo code, all the N_r rounds are identical with the exception of the final round which does not include the *MixColumns* transformation. The array werepresents the round keys that are generated by the key expansion routine. In the following sections, individual transformations that are used in each encryption round are described.Cipher(byte PlainText[4*N_b], byte CipherText[4*N_b], word w[N_b*(N_r+1)])

```
begin
  byte state[4,Nb]
  state = in
AddRoundKey(state, w[0, Nb-1])
  for round = 1 step 1 to Nr–1
SubBytes(state)
ShiftRows(state)
MixColumns(state)
AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
  end for
SubBytes(state)
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
out = state
end
Cipher(byte PlainText[4*Nb], byte CipherText[4*Nb], word
w[Nb*(Nr+1)])
begin
  byte state[4,Nb]
  state = in  AddRoundKey(state, w[0, Nb-1])
  for round = 1 step 1 to Nr–1
SubBytes(state)
ShiftRows(state)
MixColumns(state)
AddRoundKey(state, w[round*Nb, (round+1)*Nb-1]) end
forSubBytes(state)
ShiftRows(state)
AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
out = state
end
```

Fig 3 – AES(Advanced Encryption Standard) Cipher

TPS

TPS is nothing but a Transaction Processing System which gives an assurance scalable operation between client servers as well distributed architecture. Basically in our project we have use TPS concept for data reliability purpose.

In a given research work when data will get divided in different blocks; we have to store it scheduled different cloud servers. The given schema gives an assurance to end user his data is scalable and integrated. We consider if we have two different cloud servers then how do we use TPS at addition time. We will use the transaction like this

1. Begin the transaction
2. Execute a set of data manipulations and/or queries
3. If no errors occur then commit the transaction and end it
4. If errors occur then rollback the transaction and end it

If no errors occurred during the execution of the transaction then the system commits the transaction. A transaction commit operation applies all data manipulations within the scope of the transaction and persists the results to the cloud database. If an error occurs during the transaction, or if the user specifies a rollback operation, the data manipulations within the transaction are not persisted to the database. In no case can a partial transaction be committed to the database since that would leave the database in an incompatible state.

VI. CONCLUSION

It is clear that although the use of cloud computing has rapidly increased; cloud computing security is still considered the major issue in the cloud computing environment. Customers do not want to lose their private information as a result of malicious insiders in the cloud. In addition, the loss of service availability has caused many problems for a large number of customers recently. In addition, data intrusion leads to many problems for the users of cloud computing. The purpose of this project is to implement secured-clouds to address the security risks and solutions. By using AES algorithmand scalability of transaction of cloud system.

ACKNOWLEDGMENT

I take this opportunity to thank my internal guide **Prof. RekhaKulkarni** for giving me guidance and support throughout the dissertation. Her valuable guidelines and suggestions were very helpful.
I am also grateful to **Prof. G. P. Potdar,** Head of Computer Engineering Department, Pune Institute of Computer Technology for giving me all the help and important suggestions all over the dissertation.
I also thank to **Dr. P. T. Kulkarni**, Principal,Pune Institute of Computer Technology for his encouragement and providing the required resources.

REFERENCES

[1]AbhaSachdev ,MohitBhansali "Enhancing Cloud Computing Security using AES Algorithm "
[2] A. Bessani, M. Correia, B. Quaresma, F. Andre and P. Sousa "DepSky: dependable and secure storage in a cloud-of-clouds ", EuroSys'11:Proc. 6thConf. On Computer systems, 2011, pp. 31-46.
[3] CongWang, KuiRen, and JiaWang "Secure and Practical Outsourcing of Linear Programming in Cloud Computing ", IEEE transactions on cloud computing April 10-15, 2011
[4] Zhou Wei, Guillaume Pierre, Chi-Hung Chi "CloudTPS: Scalable Transactions for Web Applications in theCloud", IEEE transactions on services computing, special issue on cloud computing, 2011
[5]http://en.wikipedia.org/wiki/Shamir's_Secret_Sharing
[6]K.D. Bowers, A. Juels and A. Oprea, "HAIL: A high-availability and integrity layer for cloud storage", CCS'09: Proc. 16th ACM Conf. onComputer and communications security, 2009, pp. 187-198.
[7]C. Cachin, I. Keidar and A. Shraer, "Trusting the cloud", ACM SIGACT News, 40, 2009, pp. 81-86.
[8]R.C. Merkle, "Protocols for public key cryptosystems", IEEE Symposium on Security and Privacy, 1980, pp. 122-134.
[9] H. Abu-Libdeh, L. Princehouse and Weatherspoon, "RACS: a case for cloud storage diversity", SoCC'10:Proc. 1st ACM symposium onCloud computing, 2010, pp. 229-240.
[10] A. Bessani, M. Correia, B. Quaresma, F. André and P. Sousa, "DepSky: dependable and secure storage in a cloud-of-clouds", EuroSys'11:Proc. 6thConf. On Computer systems, 2011, pp. 31-46.
[11]Qian Wang, Cong Wang, KuiRen, Wenjing Lou and Jin Li "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing "
[12]FarzadSabahi Faculty of Computer Engineering Azad University Iran fsabahi@ieee.org "Cloud Computing Security Threats and Responses "
[13]Navia Jose1, Clara Kanmani A2 "Data Security Model Enhancement In Cloud Environment".
[14] Mohammed A. AlZain , Eric Pardede , Ben Soh , James A. Thom "Cloud Computing Security: From Single to Multi-Clouds", 45th Hawaii International Conference on System Sciences,2012.
[15] AmanpreetKaur, Gaurav Raj "Secure Broker Cloud Computing Paradigm sing AES and Selective AES Algorithm"

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. 2, Issue 11, November 2014