Journal of Computer Science 2012, 8 (12), 1987-1995
ISSN 1549-3636

doi:10.3844/jcssp.2012.1987.1995 Published Online 8 (12) 2012 (http://www.thescipub.com/jcs.toc)

DATA INTEGRITY PROOF AND SECURE COMPUTATION IN CLOUD COMPUTING

[1]Veeralakshmi Ponnuramu and [2]Latha Tamilselvan

[1]Department of Computer Science and Engineering,
[2]Department of Information Technology,
School of Computer and Information Sciences,
B.S. Abdur Rahman University, Chennai, Tamilnadu, India

Received 2012-08-09, Revised 2012-08-27; Accepted 2012-12-05

ABSTRACT

Cloud computing is an emerging computing paradigm in which information technology resources and capacities are provided as services over the internet. The users can remotely store their data into the cloud so that the users can be relieved from the burden of local data storage and maintenance. The user does not have any control on the remotely located data. This unique feature possess many security challenges. One of the important concern is the integrity of data and computations. To ensure correctness of user's data in the cloud, an effective scheme assuring the integrity of the data stored in the cloud is proposed. We try to obtain and prove that the data stored in the cloud is not modified by the provider, thereby ensuring the integrity of data. To ensure secure computation our scheme uses the Merkle hash tree for checking the correctness of computations done by the cloud service provider. Algorithms are implemented using java core concepts and java Remote Method Invocation (RMI) concepts for client-server communication by setting up the private cloud environment with eucalyptus tool. This method is used to assure data integrity and secured computations with reduced computational and storage overhead of the client.

Keywords: Cloud Computing Security, Data Storage, Merkle Hash Tree, Commitment Generation

1. INTRODUCTION

Cloud computing is a pay-per-use model for enabling available, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. Cloud computing refers to the delivery of scalable IT resources over the internet, as opposed to hosting and operating those resources locally, such as on a college or university network. Those resources can include applications and services, as well as the infrastructure on which they operate. In cloud computing users can access storages and applications from remote cloud servers by fixed or mobile terminals. By deploying IT infrastructure and services over the network, an organization can purchase these resources on an as-needed basis and avoid the capital costs of software and hardware. With cloud computing, IT capacity can be adjusted quickly and easily to accommodate changes in demand (Armbrust *et al*., 2009).

1.1. Cloud Computing Services Can Be Classified into Three Services

Infrastructure As A Service (IAAS), Platform As A Service (PAAS) and Software As A Service (SAAS). Infrastructure as a Service (IaaS) involves outsourcing the equipment used to support operations, including storage, hardware, servers and networking components. Platform as a Service (PaaS) is a paradigm for delivering operating systems and associated services over the internet without downloads or installation i.e., the development environment is offered as a service. Software As A Service (SAAS) is a software distribution model in which applications are hosted by a vendor or service provider and made available to customers over a network, typically the internet. The cheap and powerful processors, together with the "Software as a Service"

Corresponding Author: Veeralakshmi Ponnuramu, Department of Computer Science and Engineering,
B.S. Abdur Rahman University, Chennai, Tamilnadu, India

(SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable flexible network connections make it possible for clients to subscribe high-quality services from data and software that reside solely on remote data centers.

There are numerous security and privacy (Pearson, 2009) issues for cloud computing as it encompasses many technologies including networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing and memory management. These issues fall into two broad categories-security issues faced by cloud providers and security issues faced by their customers. In most cases, the organizations providing software, platform, or Infrastructure as-a-Service via the cloud must ensure that their infrastructure is secure and that their clients data and applications are protected. The customer must also ensure that the provider has taken the proper security measures to protect their information. Cloud computing moves the application software and databases to the large data centers, where the management of data and services may not be trustworthy. This unique attribute possess many new security challenges. The world of cloud computing offers many benefits like limitless flexibility, better reliability, enhanced collaboration, portability and simpler devices. To enjoy the full benefit of cloud computing, we need to address the privacy and security concerns. In this study, the cloud security is divided into two classes.

1.2. Stored Data Integrity

It refers to ensuring the integrity of outsourced data stored at the untrusted cloud servers. In this we deal with the problem of implementing a protocol for obtaining a proof of data possession in the cloud. This problem tries to obtain and verify a proof that the data that is stored by a user at a remote data storage in the cloud is not modified by the archive and thereby the integrity of data is assured. This verification system prevent the cloud storage archives from misrepresenting or modifying the data stored in it without the consent of the data owner by using frequent checks on the storage archives.

1.3. Cloud Computation Security

It refers to checking the result of the outsourced computation by untrusted cloud servers. The cloud user submits many tasks and data to the cloud server for computation. The cloud server could cheat the cloud users in two ways:

- The cloud server computes some functions and return the cloud users a random number instead, but claims to have completed all the computations
- The cloud server chooses some wrong data which has much lowest computational cost and claims to use the correct data while the original data is missing. In this study, a scheme using Merkle hash tree to detect the cheating behavior of cloud service provider is proposed

2. MATERIALS AND METHODS

2.1. Cloud Security Issues

Recently, much of growing interest has been pursued in the context of remotely stored data verification. Some security issues arising from the usage of cloud services and by the underlying technologies used to build the cross-domain internet-connected collaborations are discussed in (Jensen *et al*., 2009). It focuses on WS-security, transport layer security, browser security, cloud integrity and binding issues.

2.2. Merkle Hash Tree

Wang *et al*. (2011) allows some third parity auditor, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand. Using Merkle hash tree it also allows the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. In this, the third party verifier can misuse the data while they are doing the verification operation. Lifei *et al*. (2010) proposed a mechanism for checking the correctness of computations done by the cloud service provider. In this, they have used the Merkle hash tree to check the correctness of the computation. The drawback in this scheme is, the number of computations the cloud user submits to the provider must be in the power of 2, since the Merkle hash tree can be constructed for the number of nodes of power 2.

2.3. Pre-Computed Tokens

Wang *et al*. (2009a; 2009b) defined a storage correctness model, for ensuring the correctness of stored data. Their scheme relies on precomputed tokens. The user pre-computes a certain number of short verification tokens, each covering a random subset of data blocks. The cloud user challenges the cloud server with a set of randomly generated block indices. Upon receiving the challenge, each cloud server computes a short signature over the specified blocks and returns them to the user. The values of the signatures must match the corresponding tokens pre-computed by the user. The main drawback in this scheme is, the cloud user can able to challenge the cloud server only a specified number of times.

2.4. Proof Of Retrievability (POR)

Juels and Kaliski (2007) uses some sentinel characters embedded in the data file for checking the integrity. The sentinels are hidden among other blocks in the data file F. In the verification phase, to check the integrity of the data file, the verifier challenges the provider by specifying the positions of a collection of sentinels and asking the provider to return the associated sentinel values. In this scheme, the cloud user has to note the positions of the sentinel values and the number of times that the cloud user challenging the cloud server is also limited. Ateniese *et al.* (2007) defined "Provable Data Possession" model for ensuring possession of files on untrusted storages. In their scheme, they utilize RSA- based homomorphic tags for auditing outsourced data. In this the cloud user has to pre-compute the tags and store all the tags. This tags need a lot of computation and storage space. Shacham and Waters (2008) used the homomorphic properties for checking the integrity of data. Chang and Xu (2008) used the MAC and reed solomon code for checking the remote integrity. The homomorrphic properties, MAC and reed solomon code cannot be applied for checking the correcteness of computations.

2.5. Problem Statement

One of the important concern is the integrity of data and computation. Providers must ensure that all critical data are masked and only authorized users have access to data in its entirety. Cloud providers must also ensure that applications available as a service via the cloud are secure .We consider a general cloud computing model consisting of n cloud servers, S1,S2,..Sn, which may be under the control of one or more Cloud Service Providers (CSP). The cloud user stores the data in the cloud servers. The cloud user uses the cloud servers for data storage and submits some tasks for computation. The cloud service provider can compromise the user in two ways.

2.6. Storage Misuse

The Cloud Service Providers (CSPs) might delete some rarely accessed data files to reduce the storage cost or modify the stored data to compromise the data integrity.

2.7. Compromising Computation

The cloud user submits many tasks and data to the cloud server for computation. The cloud server could cheat the cloud users in two ways. (i)The cloud users computes some functions and return the cloud users a random number instead, but claims to have completed all the computations. (ii) The cloud server chooses some wrong data which has much lowest computational cost and claims to use the correct data while the original data is missing. In this study, a scheme using Merkle hash tree to detect the cheating behavior of cloud service provider is proposed.

2.8. Proposed Algorithm

To ensure correctness of user's data in the cloud, an effective scheme is proposed with two salient features:

- Obtain and verify a proof that the data stored in the cloud is not modified by the provider, thereby the integrity of data is assured
- To ensure secure computation our scheme uses the Merkle hash tree for checking the correctness of computations done by the cloud service provider

2.9. Ensuring Data Integrity

This verification system prevent the cloud storage archives from misrepresenting or modifying the data stored in it without the consent of the data owner by using frequent checks on the storage archives. For checking the integrity of data, first generate meta-data for each data block in the file and append it to the original data. Store this meta-data along with the original data in the cloud server. When the verifier wants to verify the integrity of the file F, the user throws a challenge to the server and asks the server to respond. The challenge specifies the block number and the byte number in the data block that has to be verified. The server responds with two values (i) the value of meta-data and (ii) the value of original data. The verifier decrypts the metadata and verifies if the decrypted value is the same as the value of the original data. If the values are same then integrity is assured. The communication between the cloud server and user is depicted in the **Fig. 1**.

2.10. Algorithm for Generating the Meta-Data

- Split the datafile F into n data blocks d1, d2, d3,.. dn
- Let each of the n data blocks contains m bytes like b1, b2, b3,.. bm
- For every data blocks in the data file F, generate the metadata by using the function Eq. 1:

$$f(i,j) = i * j * \text{ASII}(\text{char}[i,j]) \quad .i = 1,2,3,..n;\ j = 1,2,3,..m \tag{1}$$

f(i,j)-> refers to the j'th byte in the I'th block

- Append the metadata value to the original data
- Store the appended meta-data and original data into the cloudserver

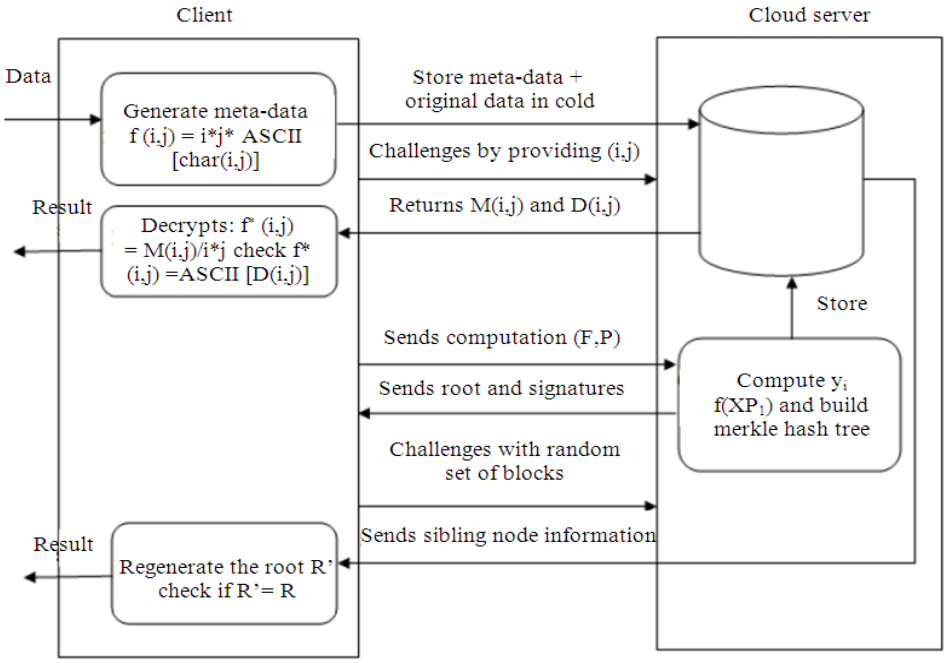

Fig. 1. System flow diagram

2.11. Algorithm for Checking the Data Integrity

- The verifier challenges the cloud storage server by specifying the block number i and the byte number j. So the verifier sends an message like challenge (i,j) to the cloud server
- The cloud server looks for the j'th data byte in the I'th data block, in both the meta-data block and in the original data block. The cloud server sends two values M (i,j) and the D (i,j) to the verifier M (i,j)->value of meta-data at the jth byte in the I'th block D (i,j)->value of original-data at the jth byte in the I'th block
- The verifier do the inverse function Eq. 2:

$$f'(i,j) = M(i,j) / (i * j) \quad (2)$$

- If the equation 3 holds, then the data is not modified Eq. 3:

$$f'(i,j) = ASII(D(i,j)) \quad (3)$$

- If the equation.3 does not hold, then the data is modified. From the steps 3 and 4 of the data integrity checking algorithm, the modification of stored data has been detected thereby assuring the data integrity

2.12. Ensuring Secured Computation

Merkle Hash Tree (MHT) is a well known authentication structure proposed by Merkle, which is constructed as a binary tree where each leaf of the tree is a hash value of authentic values. It is used to ensure the authenticity and integrity.

In this proposal, Merkle hash tree is used to ensure the correctness of computations done by the cloud server. It is based on the Merkle hash tree commitment scheme which includes the following procedures:

- Computation commitment generation
- Computation verification

2.13. Computation Commitment Generation

The cloud server is generating the Merkle hash tree as commitment to be given to the cloud user. It is generated using the following steps:

- The cloud user submits a number of computational service requests to the service provider i.e., a set of functions F = {f1,f2,…fn} over the data blocks P = {p1,p2,..pn}
- When the cloud server receives the computing requests {F,P}, it inputs the data in the position P, computes each function as yi = fi(Xpi) and the builds the Merkle hash tree

- The cloud server constructs n leaves with the values {Vi = H(yi||pi)}. Then the cloud server builds the complete Merkle tree using these leaf values from bottom to top, where value of internal node is the combined hash function of the left and right child. In this manner, the root R of Merkle hash tree **Fig. 2** is obtained
- The cloud server signs the root R and generate a signature Sig (R) and sends the computational results and the signature Sig (R) to the cloud users. The users uses the Sig (R) to verify the computation results

Usually, Merkle hash tree can be generated for the number of leaves in order of power of 2. So, the number of requests for computations from the cloud user must be power of 2. To avoid this difficulty dummy nodes are inserted to make the number of computations as a power of 2 in **Fig. 3**.

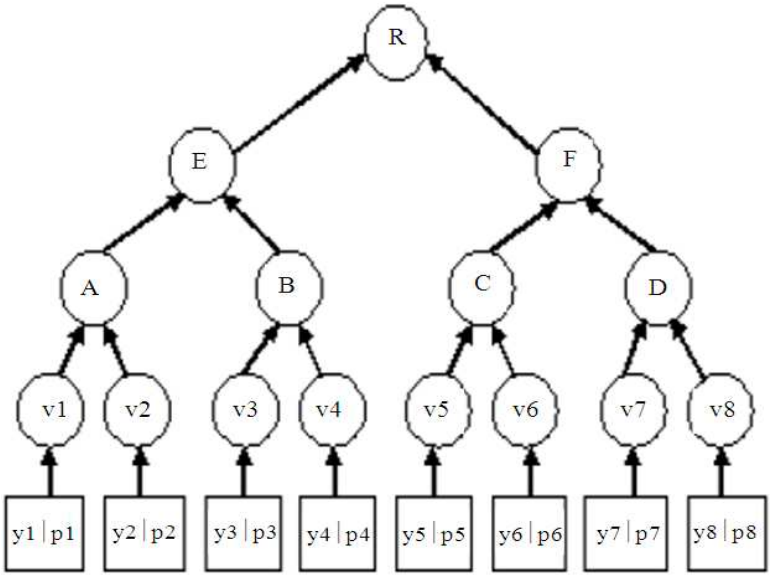

Fig. 2. Merkle hash tree built by the cloud server

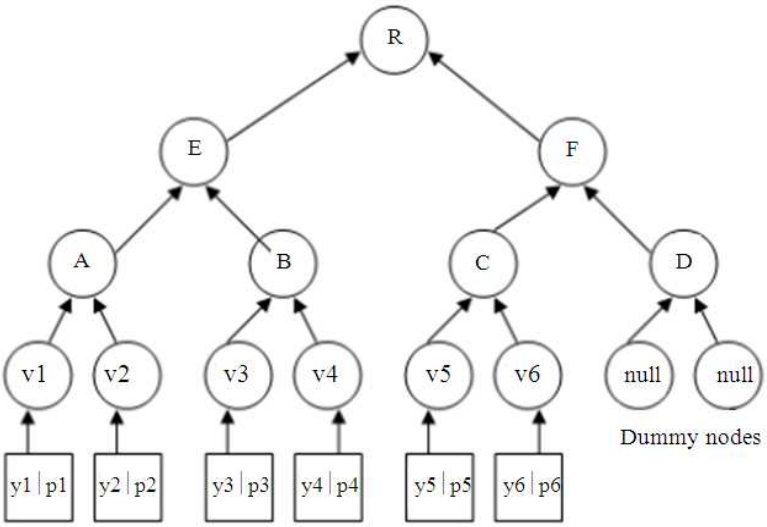

Fig. 3. Merkle hash tree with inserted dummy nodes

2.14. Computation Verification

The cloud user does the computation verification by using the following steps:

- The cloud user performs verification by selecting a random subset S = {c1, c2, ..cn} from the domain [1,n]. and sends this challenge request to cloud server
- For each ci Є S, the cloud server finds in the Merkle hash tree a path Φci, from the leaf to the root. For each node on this path Φci, cloud server sends the sibling sets to cloud user. For example, the challenge on f4(x4) needs to compute a path Φ4 with the vertices {v4, B, E, R}. To perform this computation each node"s sibling vertices is required to compute the root R. So the cloud server returns the values X4, Sig (R) and the value set {v3, A, F} back to the challenger
- The cloud user gets the values from the cloud server and generates the signature Sig'(R) using the result and the sibling value set sent by the cloud server. If the signature Sig'(R) matches with the Sig (R), the cloud user confirm that the computations are done correctly

3. RESULTS

A private cloud environment is deployed using the eucalyptus tool which is provided along with the Ubuntu Enterprise Cloud (UEC). UEC is a stack of applications from Canonical included with Ubuntu server edition. UEC includes eucalyptus along with a number of other open source software. UEC makes it very easy to install and configure the cloud. Eucalyptus is a software platform for the implementation of private cloud computing on computer clusters. It provides an C2-compatible cloud computing platform and S3-compatible cloud storage platform. Eucalyptus works with most currently available Linux distributions including Ubuntu, Red Hat Enterprise Linux, CentOS, SUSE Linux Enterprise Server, openSUSE, Debian and Fedora. It can also host Microsoft Windows images. Eucalyptus is an acronym for "Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems". To install and configure a basic UEC three systems are required. Two servers (server1 and server2) will run 32-bit server version and the third server will run a desktop 32-bit version (Client1). The desktop version of Ubuntu is installed on client1 so that firefox or other browsers can be utilized to access the web interface of UEC. Our experiment is conducted on three systems with the configurations listed in **Table 1**.

Table 1. Hardware configurations

Hardware	Server1	Server2	Client1
CPU	1 GHz	VT extensions	VT extentions
RAM	1 GB	1 GB	1 GB
Disk	5400 rpm IDE	5400 rpm IDE	5400 rpm IDE
Disk space	40 GB	40 GB	40 GB
Networking	100 Mbps	100 Mbps	100 Mbps

Table 2. Generated meta data values for the text "cloud computing"

Character	Blockno (i)	Blockno (j)	Ascii (i,j)	Meta data
C	1	1	99	99
L	1	2	108	216
O	1	3	111	333
U	1	4	117	468
D	2	1	100	200
Space	2	2	30	120
C	2	3	99	594
O	2	4	111	888
M	3	1	109	327
P	3	2	113	678
U	3	3	117	1053
T	3	4	116	1392
I	4	1	105	420
N	4	2	110	880
G	4	3	103	1236

Algorithms are implemented using java core concepts and we have used java Remote Method Invocation (RMI) concepts for client-server communication.

3.1. Storage Management

Walrus is a storage service in eucalyptus which is compatible with Amazon's S3 (Simple Storage Service). Using walrus the users can store persistent data, which is organized as buckets and objects. WS3 is a file level storage system, as compared to the block level storage system of storage controller. Walrus controller options can be modified from the Web UI, on the\Configuration" page under\Walrus Configuration" section. For using walrus to manage eucalyptus Virtual Machine (VM) images, Amazon's tools are used to store/register/delete them from walrus. Other third party tools can also be used to interact with walrus directly. Some of third party tools for interacting with walrus are:

- s3curl-S3 Curl is a command line tool that is a wrapper around curl
- s3cmd-is a tool that allows command line access to storage that supports the S3 API
- s3fs-is a tool that allows users to access S3 buckets as local directories

S3 Curl is used to interact with walrus for storing data in the server. Users may create, delete, list buckets, put, get, delete objects, set access control policies, with S3 Curl tool. A perlscript called s3curl.pl from Amazon is used to create buckets in the Walrus and store data in the bucket.

For the text file containing the text "cloud computing" we generated the metadata as follows.

We assume these parameters:

$$n = 4, F = data.txt, text = "could computing"$$

The file F is split into 4 blocks of 4 bytes each. The metadata is generated using the equation.1. The generated meta value for each byte is displayed in the **Table 2**.

3.2. Example 1

After appending metadata to the original data the file looks as shown in the **Fig. 4**.

A bucket is created in walrus storage area and metadata.txt file is stored in the bucket. To check the integrity of this file a challenge (1,4) is sent to the server. The server returns M(1,4) as 468 and D(1,4) as „u". Using the equation 2, the inverse function f'(i,j) is calculated:

$$f'(1,4) = 468 / (1*4) = 117 = 1, j = 4$$

117 is the ASCII (u) and D (1,4) returned from the server is also 'u'. So the equation. 3 f' (1,4) = d(1,4) = 117 holds.

3.3. Example 2

Suppose the character at the positions (3,1), (3,2) are modified to 'c' instead of 'm' and 'a' instead of the character 'p' the metadata.txt file looks as ashown in the **Fig. 5**.

To check the integrity of this file a challenge (3,2) is sent to the server. The server returns M (3,2) as 678 and D (3,2) as „a". Using the equation 2, the inverse function f'(i,j) is calculated:

$$f'(3,2) = 678 / (3*2) = 113 i = 3, j = 2$$

113 is the ASCII (p) and D (3, 2) returned from the server is 'a'. Now f'(3,2)=113 and ASCII(a) = 97. Here the equation.3 does not hold. From the example 1 and 2, it is concluded that modification of data can be detected. So from this, our proposed algorithm for checking data integrity has been proved.

3.4. Example 3

Consider the case that, the proposed algorithm has not been applied to the data and the plain data is stored as such in the server as shown in the **Fig. 6**.

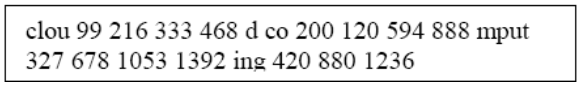

Fig. 4. Metadata.txt

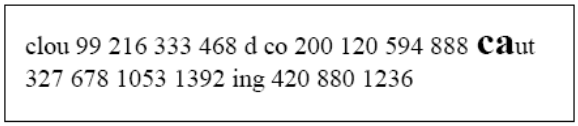

Fig. 5. Modified metadata.txt

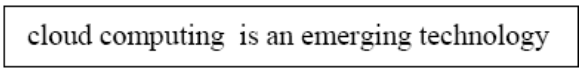

Fig. 6. Data txt

If the text in the **Fig. 6** is modified as "cloud computing is not an emerging technology" and when the user retrieves the file, the verifier can read only the modified text without knowing the modification of the text. In order to avoid this difficulty, our integrity checking algorithm can be used.

For checking the computation integrity, request consisting of some functions like addition, multiplication, maximum, minimum, average is given to the server. The computation results and signature are received from the server. The computation results and signature is verified and security of computation is assured.

4. DISCUSSION

4.1. Computation and Storage Cost

The client generates the meta data, encrypt the meta-data and append the data to the original data and store the data at the server. This incurs some extra computation cost in the client side. After the computation, the size of the file is doubled. So the client has to get double the file size of storage space from the cloud service provider. Even though the storage cost is increasing, the integrity of data stored in the cloud server is assured here. The comparison of file sizes for the original data and metadata is depicted in the **Fig. 7**.

Data security risk stems primarily from loss of physical, personnel and logical control of data. Issues include virtualization vulnerabilities (STA, 2008), SaaS vulnerabilities (e.g., a case in which Google Docs exposed private user files) Google Docs Glitch Exposes Private Files, 2011, phishing scams (McMillan, 2007) and other potential data breaches.

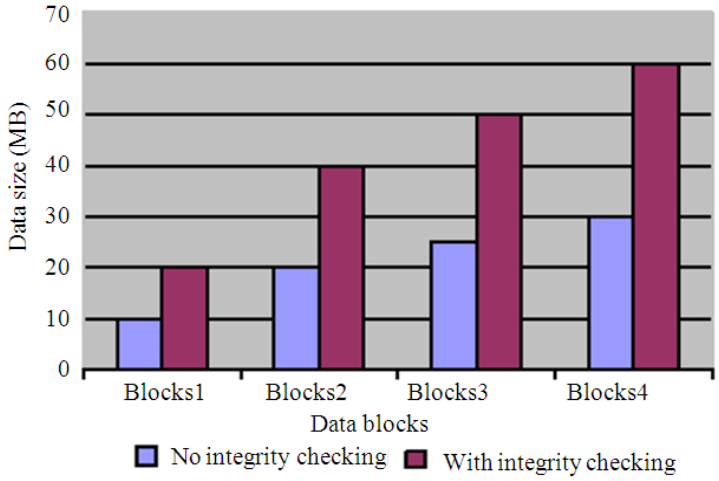

Fig. 7. Comparison the data size of files with meta- data and without meta-data

Table 3. Probability of integrity assurance for the data stored in the cloud serve

Files stored in the cloud server	Data stored as plain text (without metadata)	Data stored in the server along with metadata
File 1	0.590	1
File 2	0.634	1
File 3	0.950	1
File 4	0.204	1
File 5	1.000	1
File 6	0.500	1

Other data security risks mentioned in (Catteddu and Hogben, 2009) include data leakage and interception, economic and distributed denial of service and loss of encryption keys. Unique risks also arise due to the multi-tenancy and resource-sharing models. The inability to fully segregate data or isolate separate users can lead to undesired exposure of confidential data in the investigation of a situation involving co-tenants. Hypervisor vulnerabilities can also be leveraged to launch attacks across tenant accounts. Data containing social and national insurance details, health data and financial information raise issues about authorization, rights management, authentication and access controls.

After the detailed analysis, it is found that only a small percentage of files stored in the cloud server is integrity assured if the data is stored as the plaindata (Jensen *et al*., 2009). From the example 3, the probability of data integrity assurance is assumed as shown in the **Table 3**. But our proposal is giving 100% assurance for all the files stored in the cloud server.

5. CONCLUSION

In this study a method for checking the integrity of stored data and the correctness of computations done by the cloud server is proposed. This scheme is introduced to reduce the computational and storage overhead of the client. The main advantage of this method is that, storage at the client side is minimal, because the client has to remember only two functions f (i,j) and f'(i,j). This method works only to static storage of data. It cannot handle the case when the data need to be dynamically changed. Future works may be concentrated on working with dynamically changing data.

6. REFERENCES

Armbrust, M., F. Armando, R. Griffith, D. Anthony and R. Joseph *et al*., 2009. Above the clouds: A Berkeley view of cloud computing. University of California, Berkeley.

Ateniese, G., R. Burns, R. Curtmola, J. Herring and L. Kissner *et al*., 2007. Provable data possession at untrusted stores. Proceedings of the 14th ACM Conference on Computer and Communications Security, Oct. 28-31, ACM Press, New York, pp: 598-609. DOI: 10.1145/1315245.1315318

Catteddu, D. and G. Hogben, 2009. Cloud computing risk assessment. Eur. Netw. Inform. Security Agency.

Chang, E.C. and J. Xu, 2008. Remote integrity check with dishonest storage server. Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security, (ESORICS' 08), ACM Press, Heidelberg, pp: 223-237. DOI: 10.1007/978-3-540-88313-5_15

Jensen, M., J. Schwenk, N. Gruschka and L.L. Iacono, 2009. On technical security issues in cloud computing. Proceedings of the IEEE International Conference on Cloud Computing, (CLOUD-II 2009), Sept. 21-25, IEEE Xplore Press, Bangalore, pp: 109-116. DOI: 10.1109/CLOUD.2009.60

Juels, A. and B.S. Kaliski, 2007. Pors: Proofs of retrievability for large files. Proceedings of the 14th ACM Conference on Computer and Communications Security, (CCS' 07), ACM Press, New York, pp: 584-597. DOI: 10.1145/1315245.1315317

Lifei, W., H. Zhu, C. Zhenfu and W. Jia, 2010. SecCloud: Bridging secure storage and computation in cloud. Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems Workshops, Jun., 21-25, IEEE Xplore Press, Genova, pp: 52-61. DOI: 10.1109/ICDCSW.2010.36

McMillan, R., 2007. Salesforce.com warns customers of phishing scam.

Pearson, S., 2009. Taking account of privacy when designing cloud computing services. Proceedings of the ICSE Workshop on Software Engineering Challenges of Cloud Computing, (CLOUD' 09), ACM Press, USA., pp: 44-52. DOI: 10.1109/CLOUD.2009.5071532

Shacham, H. and B. Waters, 2008. Compact proofs of retrievability. Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, (ASIACRYPT' 08), ACM Press, Heidelberg, pp: 90-107. DOI: 10.1007/978-3-540-89255-7_7

STA, 2008. VMware shared folder bug lets local users on the guest OS gain elevated privileges on the host OS.

Wang, C., Q. Wang, K. Ren and W. Lou, 2009a. Ensuring data storage security in cloud computing. Proceedings of the17th International Workshop on Quality of Service, Jul. 13-15, IEEE Xplore Press, Charleston, SC., pp: 1-9. DOI: 10.1109/IWQoS.2009.5201385

Wang, Q., C. Wang, J. Li, K. Ren and W. Lou, 2009b. Enabling public verifiability and data dynamics for storage security in cloud computing. Proceedings of the 14th European Conference on Research in Computer Security, (ESORICS' 09), ACM Press, Heidelberg, pp: 355-370.

Wang, Q., C. Wang, J. Li, K. Ren and W. Lou, 2011. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parall. Distrib. Syst., 22: 847-858. DOI: 10.1109/TPDS.2010.183