Algorithmic Solution to Reduce Storage Overhead On Cloud Based System

Ms. Dhanashri P. Joshi[1], Prof. N. P. Kulkarni[2]

[1]Dept. of Information Technology
Sinhgad Technical Education Society's SKNCOE, Pune, India

[2]Dept. of Information Technology
Sinhgad Technical Education Society's SKNCOE, Pune, India

Abstract

Cloud computing is a technology that provide on-demand services namely Storage-as-a-Service, Platform-as-a-Service etc. over the internet via Cloud vendors from the market to the user. Cloud computing allows utilization of the services regardless of the installation and maintenance problems to the users. Storage of data on single local machine or server results into the maximization of storage cost and maintenance cost. So these days most of the organizations choose the option to store such data on a cloud server. Cloud provides Storage-as-a-Service where organization can use the space on a cloud server to store data. While storing, there may occur some storage overhead. So there are some challenges such as to reduce storage overhead on the cloud server. In this paper, a novel methodology is introduced that will provide a solution for reducing storage overhead on cloud. There is approximately 0.3% storage overhead in existing system (That means 0.3% extra space required to store the file) whereas in proposed system there is 92.1% reduction in requirement of space.

Keywords: *cloud computing, Storage as a Service, Storage overhead.*

1. Introduction

Cloud Computing is a technology that provide on-demand services namely Storage-as-a-service, Infrastructure-as-a-service, etc. over the internet via Cloud vendors to the organization. Cloud computing gives the ability to the organization or individual user for utilizing the services provided by cloud regardless of the problems in installation and maintenance. The authorized users of cloud services could be able to access the data stored on the cloud. When cloud is used to store dynamic data, there is some amount of storage overhead occurs. This paper is giving novel idea to reduce this storage space requirement; for this purpose one of the lossless compression technique is used.

In this paper Section 1 contains basic information about domain and introduction of the problem and its solution. Section 2 contains an overview of existing systems. Section 3 includes a description of system architecture. Section 4 contains an explanation about proposed system which will be having less storage overhead. Section 5 describes the implementation and its settings. Section 6 contains the results. Section 7 contains the conclusion of this system and also what enhancement we can do in this.

2. Related Work

A. Lazy Revocation

In traditional cryptographic systems, keys are different for each file. So it needs to maintain many keys. PLUTUS is system in which each group of files is associated with the key. So number of keys get reduced. This system uses key-updating scheme and improves key rotation mechanism in the file system. Lazy revocation requires fewer keys. So this is very efficient key management method and less expensive too [3].

B. Broadcast Encryption

This system encrypts message for system users who are listening on the broadcast channel. Any user from those can use a private key to decrypt message. It not only worked on random oracle model but also on the actual system[5].

C. Trusted Third Party

TTP is trusted third party verifier. TTP is an actor in the system on which both client and CSP trust. New system was introduced which is a combination of identity-based cryptographic technique and RSA digital signature. This is only securely work on random oracle model. Random oracle model is a model gives response to every request but with random output and responses same for the same query every time [6].

TTP is used in RSA based assumption data integrity check. But this method is proposed for dynamic data. For enabling latest version of data make available for authorized user system has key-rotation. System divides file into blocks and block status table is used similarly to SN-BN table[1]. There are numbers of compression algorithms that are of type lossless compression. But from all of them comparatively Deflate algorithm is more efficient. Around 90% storage space can be reduced using deflate [4].

3. System Architecture

Assumptions & System Model

In cloud-based storage model, there are four actors' as shown in Fig. 1. They are data owner, cloud service provider, trusted third party and authorized user.

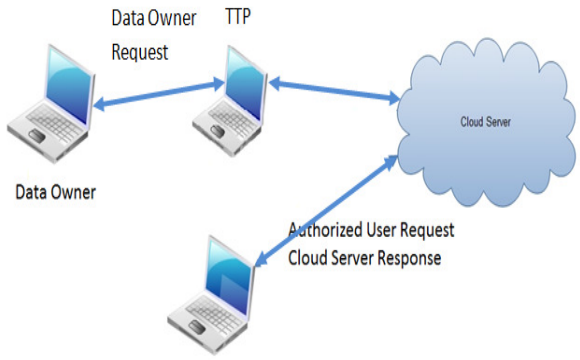

Fig. 1: Architecture of proposed system

Data owner is an organization or an individual who is generation sensitive data and want to store it on a cloud server. Data owner can do all the block level operations on data as insertion or deletion. Data owner does not directly trust Cloud Service Provider (CSP).

CSP manages cloud servers. He provides free or payable service to users or data owners. He also provides space to store data.

Authorized users have rights to access data uploaded on the cloud server by data owner. He cannot do block level operations on data.
Trusted Third Party (TTP) is organization or individual. CSP and data owner does not trust each other. They both trust on TTP. So TTP is party that creates indirect mutual trust between CSP and Data owner.

Here in Fig.1 relations between all the four entities are shown by double sided arrow because there is request-response relation between all the four. Assume Data owner want to upload file F on cloud. He divides file in 'm' blocks and encrypts file block-wise for confidentiality of data. He can do all the block level operations on individual blocks. Here our constraint is that the original file should be protected from CSP, TTP and unauthorized users.

4. With encryption with compression system

4.1 System Description:

It allows the owner to outsource confidential data to a CSP, and perform operations on the block of outsourced data. It ensures that authorized receive the latest version of the outsourced data. In this paper, a novel system is proposed to implement compression and decompression technique for data upload and data download as shown in Fig.2.
In this proposed system, data owner can outsource his data. Dynamic data will get stored on the cloud server in encrypted and compressed format. File first divided into block and after that each block will encrypt and compress. Data owner can do each block level operation. Authorized user will get latest version of data when he request for it. Trusted third party (TTP) is to create a trust between data owner & CSP.

4.2 Block Status Table:

Block status table is a small table in the database used to reconstruct and access file blocks outsourced to CSP. As shown in Table 1 BST contain three fields namely serial number (SN), Bloc number (BN) and key version (KV). SN is indexing to file blocks. It

indicates physical position of the block in the file. BN is a logical number of data block. KV is used to encrypt each block in data file.BST is implemented like database.

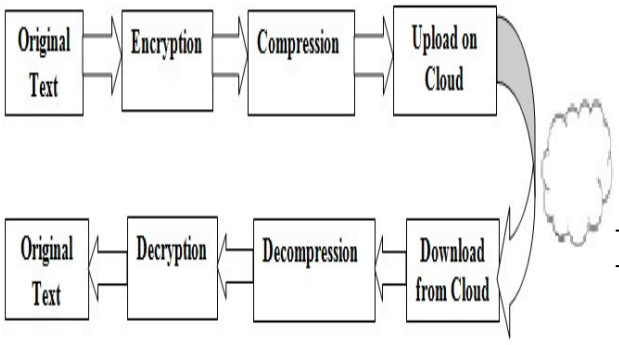

Fig. 2: Block diagram of proposed system

SN is not stored in the table and considered to be entry index[1]. BN and KV both combined require eight bytes for storage. Assume that the file is divided into m blocks then BST requires only 8m bytes of memory to store. At the time of file creation ctr (counter variable) and KV initialized to one for each block of the file. Here ctr is counter variable kept by data owner to indicate most recent version of key. When revocation is performed, ctr is incremented by 1[1].

Table 1: BST

SN	BN	KV

In this system, there are various procedural steps related to all the four users or some of them. They are as follows:

Setup at owner side:

Owner initializes ctr to 1. He generates initial key Kctr or K1, after each revocation Kctr is rotated forward. Kctr can rotate backward to enable authorized users to access blocks that are encrypted under older versions of Kctr.

Upload file:

Let us assume file F.

F is divided into n blocks.

F={blk1, blk2, blk3, ……, blkn}

Block Status Table is set of tuples that contain SN and BN for each block.

BST= {{SN1, BN1, KV1}, {SN2, BN2, KV2}, ……., {SNn, BNn, KVn}}

SNi=i, for all i=1 to n

BNi=i, for all i=1 to n

KVi=ctr, for all i=1 to n and ctr, is counter variable initialized to 1.

EK is an encryption key.

F' is the encrypted file generated by encrypting each block with BN.

blki'= EK(blki, BNi)

F'= {blk1', blk2', …. , blkn'}[2]

This encrypted file is compressed using Deflate algorithm.

File= D(F')

This file is sent to cloud server through TTP.

This file is getting uploaded on the cloud as shown in Fig. 4.

Download file:

F'= D'(File)

(blki, BNi)= EK(blki')

F={blk1, blk2, blk3, ……, blkn}[2]

Owner creates rotator (Rot) by using ctr and Kctr. Owner sends encrypted file, BST to CSP as shown in Table 2 and deletes file from local storage.

Data stored at each component of the system is as follows:

Table 2: Data stored at system component

Data Owner	TTP	CSP
ctr, K_{ctr} and BST_O	FH_{TTP}, TH_{TTP}	Encrypted & compressed file, BST_C

4.3 Operations on Dynamic Data:

Modification:

Data owner send request to CSP. If revocation done then flag set to TRUE. Assume block no. 2 is modified. blk2 is now replaced with new data block blkn+1. Initially, KV2 is 1.After revocation, KV2 is changed to 2.

Insertion:

A new value of encrypted file is calculated and sends to CSP so that he can also update his database.

Deletion:

When one block is deleted, all subsequent blocks are moved one step forward.

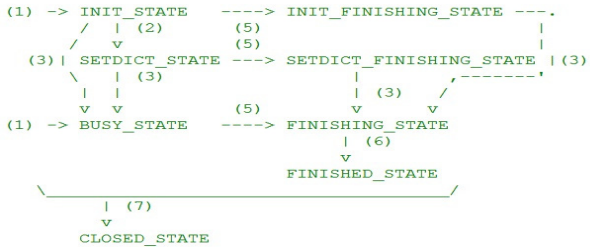

Fig. 3: Deflate state transitions

4.4 Compression Technique used in proposed system:

Compression technique used in the proposed system is Deflate. It uses a combination of two techniques LZ77 & Huffman encoding. [7] There are various deflate levels present:
The best and slowest compression level is used to find long string repetitions. [9]
The worst but fastest compression level.
The default compression level. The level won't compress at all.
The default strategy only allows long string repetitions. It is useful for random data with a small dataset.[9]
The strategy will not look for string repetitions at all. It only encodes with Huffman trees which means, that more common characters get a smaller encoding.
The Deflater can do the state transitions as shown in Fig. 3 [8]

Description of states above used is as follows

(1) If we should not produce a header we start in BUSY_STATE, otherwise we start in INIT_STATE. [8]
(2) When system is in INIT_STATE a dictionary may be set then as per indivcatwe change the state as indicated. [8]
(3) Regardlss of dictionary on the first call of deflate we change to BUSY_STATE. [8]
(4) When flush () is called to indicate that there is no more INPUT then system FINISHING_STATE.[8]
(5) When everything has been flushed system enters in FINISHED_STATE.[8]
(6) At any time (6)

5. Implementation

Proposed system is implemented on Amazon Elastic Compute Cloud (Amazon EC2). For Data owner, TTP and cloud server instances of the server are created on Amazon EC2. Windows free tier instances are used for implementation.

Implementation settings:

In an implementation ”large” Amazon EC2 instance is used to run Cloud server. Amazon cloud instance has configuration as Windows 2008sever, 64-bit base system & Instance ID: windows_server_2008-R2_SPI-English-64bit-Base_2014.05.14. One similar type of instance is used for TTP. Data owner & authorized user can login through any machine.

6. Results & Discussion

Experimental evaluation of storage overhead is given in this section. In proposed system experimental evaluation is done with taking sample file. Encryption is used which is described in existing system [1] for data security & deflate compression technique is used to reduce space required to store a file.

As shown in Fig. 4 file size increased by 0.04% after encryption, but it gets reduce by 92.04% of the original file after doing Encryption & compression successively. Fig. 5 gives a comparison between original file sizes, total space required for system storage after only encryption and total space required for system storage after encryption & compression. System storage is stored data at Data owner, TTP & CSP server as shown in Table 2.

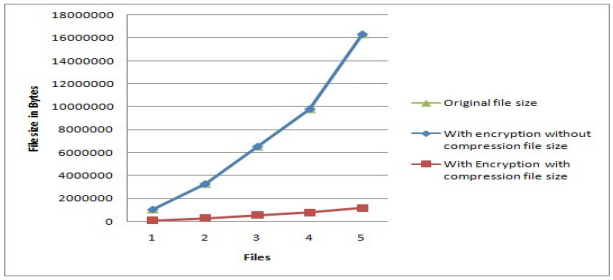

Fig. 4: Graph comparing original file, file with encryption & file with encryption with compression

After doing multiple insert, delete & modify operations on encrypted file & encrypted compressed file how much overhead is produced is shown in Fig. 6

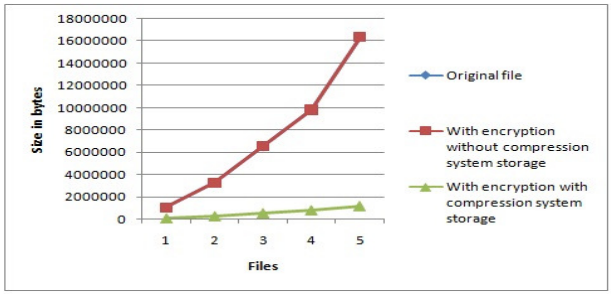

Fig. 5: Graph comparing original file, with encryption system storage & with encryption with compression system storage

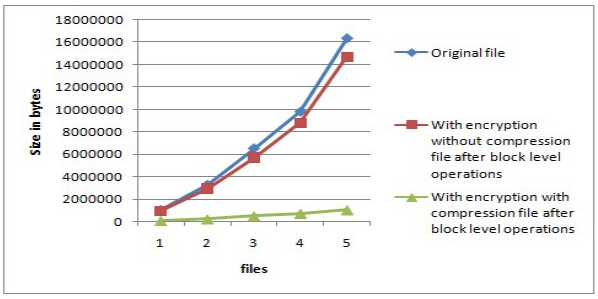

Fig. 6: Graph comparing the original file, file with encryption & file with encryption with compression after doing block level operations

7. Conclusion and Future Scope

In existing systems, there are some limitations as storage overhead, computational overhead and communication overheads. In this paper, a cloud based system is proposed which enables data owner not only to outsource his data for storage, but also to do block level operations like insert, delete & modify. Authorized user can access data blocks. TTP is working for indirect mutual trust between data owner & CSP. In existing system storage overhead is approximately 0.3%. But in the proposed system it's almost negligible.In future function for cheating detection can be implementing on TTP.

Acknowledgments

The authors would like to thank the unknown reviewers for their valuable comments and suggestions.

References

[1] Ayad Barsoum, and Anwar Hasan, "Enabling Dynamic Data and Indirect Mutual Trust for Cloud Computing Storage Systems", in IEEE transactions on parallel and distributed systems, 2012.

[2] D.P. Joshi, and Prof. N. P. Kulkarni, "A Novel Scheme to Reduce Storage Overhead on Cloud System", in Cyber Times International Journal of Technology & Management, vol. 7, Issue 1, October 2013- March 2014.

[3] Michael Backes, Christian Cachin, and Alina Oprea, "Lazy Revocation in Cryptographic File Systems", Proceedings of the Third IEEE International Security in Storage Workshop (SISW'05), 2005.

[4] S. Smyrna Grace, T. Nalini, and A. Pravin Kumar, "Secure and Compressed Secret Writing using DES and DEFLATE algorithm", in IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 2, No 3, March 2012, page no. 423-428.

[5] Wu Danfei, Zhang Weimin, "Authenticated Broadcast Encryption with Short Ciphertexts and Private Keys", in IEEE, 2011, page 218-221.

[6] Zhang lianhong, and Chen Hua, "Security Storage in the Cloud Computing: A RSA-based Assumption Data Integrity Check without Original Data", in International Conference on Educational and Information Technology (ICEIT 2010), 2010, Page 143-147.

[7] Basic deflate algorithm [onlin] Available http://www.binaryessence.com/dct/imp/en000241.htm

[8] Deflator class[online] Availab;e http://www.cis.upenn.edu/~bcpierce/courses/629/jdkdocs/api/java. util.zip.Deflater.html

[9] An explaination of deflate [online] Available http://www.cs.ucdavis.edu/~martel/122a/deflate.html

Ms. Dhanashri Joshi Birth place: Ratnagiri DOB: 28 th December 1990.Received Bachelor degree of Information Technology from Smt. Kashibai Navale College of Engineering, Vadgaon under affiliation of Pune University (2012) & pursuing Post graduate degree in Information Technology from STES, Smt. Kashibai Navale College of engineering, Pune under affiliation of Pune University. She is having 1 year of teaching experience. Research areas of interests are Cloud computing, Database.

Prof. Nandkumar Kulkarni received Bachelor of Engineering (B.E.) degree in Electronics Engineering from Walchand College of Engineering, Sangali (India) in 1996. He has been with Electronica, Pune from 1996 -2000. He worked on retrofits, CNC machines as a service engineer and was also responsible for PLC programming. In 2000, he received the Diploma in Advanced Computing (C-DAC) degree from MET's IIT, Mumbai. In 2002, he became Microsoft Certified Solution Developer (MCSD). From 2002 onwards he is working as a faculty in Pune University. From 2002 to 2005 he worked as faculty in Electronics and Telecommunications Department at GSMCOE, Pune. In 2005 he joined Electronics and Telecommunications Department at RSCOE, Pune and worked as a faculty till 2007. Since 2007, he is working with SKNCOE, Pune as a faculty in Information Technology Department. He completed Master of Technology (M.Tech) degree in Electronics Engineering from College of Engineering, Pune (India) in 2007. Currently he is a PhD scholar in Electronic Systems department at Aalborg University, Aalborg, Denmark. His area of research is Wireless Sensor Networks (WSN).