SECURE DATA COMPRESSION MODEL USING LZW IN CLOUD COMPUTING ENVIRONMENT

K. Manjupriya[1], Department of Computer Applications,
Mr. E. Ranjith[2] , MCA, M.Phil., (Ph.D), Assistant Professor,
Krishnasamy College of Engineering and Technology, Cuddalore.

Abstract

Data compression is primarily a branch of information theory which deals with techniques related to minimizing the amount of data to be transmitted and stored. The basic characteristic of data compression is to convert a string of characters into another set of characters which consists of same information but whose length is as small as possible. With the extending use of computer in various disciplines, number of data processing applications are also increasing which requires processing and storage of large volumes of data. Simultaneously, proliferation of computer networks is encouraging the passive transmission of data over communication channels. In the above described scenario, it is best to compress the data in order to reduce the storage and communication costs. Reducing the size of a file to its half is equivalent to doubling the storage medium capacity. Data compression has important application in the field of file storage and Cloud systems. It helps in reducing redundancy in stored or communicated data. This project uses LZW based compression technique to store the data efficiently in cloud environment.

Keyword: *Compressive sensing, Cloud assistance, Security, Data gathering, Encryption*

1.INTRODUCTION

Data compression is one of the enabling technologies for multimedia applications. It would not be practical to put images, audio and video on websites if do not use data compression algorithms. Mobile phones would not be able to provide communication clearly without data compression. With data compression techniques, we can reduce the consumption of resources, such as hard disk space or transmission bandwidth. Data Compression is the process of encoding data so that it takes less storage space or less transmission time. Compression is possible because most of the real world data is very redundant. In this survey, first we introduce the concept of lossy and lossless data compression techniques.

Classification of compression methods:-We have two types of compression methods:

Lossless compression: - It is used to reduce the amount of source information to be transmitted in such a way that when compressed information is decompressed, there is not any loss of information.

Lossy compression: - The aim of lossy compression is normally not to reproduce a exact copy of the information after decompression. In this case some information is lost after decompression

Lossless Compression Methods:- Run Length Encoding: - The first step in this technique is read file then it scans the file and find the repeating string of characters [6].when repeating characters found it will store those characters with the help of escape character followed by that character and count the binary number of items it is repeated. This method is useful for image having solid black pixels. This algorithm is also effective for repeating of characters. But it is not effective if data file has less repeating of characters. We can compress the run-length symbols using Huffman coding, arithmetic coding, or dictionary based methods.

Huffman Coding:- The Huffman coding algorithm is named after its inventor, David Huffman, who developed the method as a student in a class on information theory at MIT in 1950[1]. Huffman Coding Algorithm— it is a bottom-up approach 1. Initialization: Put the old nodes in a list sorted according to their frequency counts. 2. Repeat the following steps until the sorted list has only one node left: (1) From the list pick two nodes with the lowest frequency counts.

Form a Huffman sub tree that has these two nodes as child nodes and create a parent node. (2) Assign the sum of the children's frequency to the parent node and insert it into the list such that the order is maintained. (3) Delete the children from the sorted list. 3. Assign a 0 and 1codeword to the two branches of the tree on the path from the root. After the Huffman tree, the method creates a prefix code for each node from the alphabet by traversing the tree from the root to the node. It creates 0 for left node and 1 for a right node.

LZW (Lempel-Ziv Welch) compression method:- LZW is the most popular method. This technique has been applied for data compression. The main steps for this technique are given below:- Firstly it will read the file and given a code to each character. If the same characters are found in a file then it will not assign the new code and then use the existing code from a dictionary. The process is continuous until the characters in a file are null.

2. Research methodology:

The main objective of the proposed work is to improve the storage capacity in cloud and secure the data in the cloud from the unauthorized users in order to achieve better throughput and end to end delay. Figure 1 demonstrates of the general architecture for the proposed work. The proposed framework is sub divided into three stages: Requisition phase, authentication phase and storage phase. The primary stage involves the requisition phase which uses the basis login and password requisition model. The secondary stage executes authentication and authorization, whereas the authorized user have separate key to access the data in order to do that an efficient method named efficient key management authentication algorithm. In the final stage the data are compressed and stored in the network using hybrid data compression.

2.1 Efficient key management authentication algorithm:

The authorized users from the requisition phase generate a separate key. For tracing each attribute users has the unique identity. These identity and user's attributes are hiding from the users. Through this cannot learn anything from the cipher texts about the attributes matching or mismatching. The attributes are classified as the hidden normal attributes (HN) and the hidden identity attributes (HID).

A.Setup the key: This phase outputs the public key and the master key.
B.Encryption: Encrypt the message M with the set of attributes X, but the attributes are X hide hidden.
C.Key generation: Key generation can be done by access structure as input and produces the output.
D.Decryption: Decryption can be done with decryption keys for each attributes of users.

TPA (Third party Auditor) is an entity, which has expertise and capabilities for Encryption and decryption Service. When client want to store data at the cloud storage at that time TPA (encryption/decryption service) Encrypt the data and return back to user for storage purpose.
For sensitive attributes the method chooses a hash function which verifies the identity of user with the help of TPA (Third party Auditor) and once the identity verification gets cleared then the access clearance is computed. When the user request clears both the service is fulfilled and the sensitive values are encrypted using the specific key which could be decrypted by the user. For non-sensitive attributes the method uses a public key based encryption which can be decrypted by the user.

2.2 Key distribution centre:

A Distributed Key Distribution Center (DKDC, for short) is a set of n servers of a network that jointly realizes the same function as a KDC. In this setting, users have secure point-to-point channels with all servers. A user who needs to communicate with other users securely, sends a key-request message to a subset at his choice of at least k out of the n servers[13]. With this approach, the concentration of secrets and the slow down factor which arise in a network with a single KDC are removed. A single server by itself does not know the secret keys, since they are shared between the n servers. Moreover, each user can send a key-request in parallel to different servers. Hence, there is no loss of time in computing a key, compared with a centralized setting. Finally, the users can obtain the keys they need even if they are unable to contact some of the servers.[14]

2.3Hybrid data compression algorithm:

Compression technique is mainly used to reduce the space of storage and increases the capacity of the resources. The data or information which occupies more space is compressed using a compressing technique (i.e) Lossless compression technique. Then the compressed data can again be decompressed to obtain the original information for future usage. This is mainly used to reduce the resources storage space and hence increase its productivity. In this section, we are going use the Lossless data compression technique where the data or information which is compressed to minimize its storage size does not undergo any loss of data or information. The lossless compression technique is highly secured. Our work comprises of the combination of LZW algorithm and run length encoding. The LZW algorithm is very fast and simple to implement but it has the limitation of compressing the file that contain repetitive data whereas this can be overcome by run length encoding.

In this case, the encoded data consists entirely of 12 bit codes, each referring to one of the entries in the code table. In the encoding process, the cumulative probabilities are calculated and the range is created in the beginning. While reading the source character by character, the corresponding range of the character within the cumulative probability range is selected. Then the selected range is divided into sub parts according to the probabilities of the alphabet. Then the next data is read and the corresponding sub range is selected. In this way, data are read repeatedly until the end of the file is encountered. Finally a number should be taken from the final sub range as the output of the encoding process. This will be a fraction in that sub range. Therefore, the entire source message can be represented using a fraction. To decode the encoded message, the number of characters of the source message and the probability/frequency distribution are needed.
Compression is achieved by taking each code from the file, and translating it through the code table to find what character or characters it

represents. Codes 0-255 in the code table are always assigned to represent single bytes from the input file. For example, if only these first 256 codes were used, each byte in the original file would be converted into 12 bits in the LZW encoded file, resulting in a 50% larger file size. During compression, each 12 bit code would be translated via the code table back into the single bytes.

3. Performance analysis

(A) Compression ratio

Compression Ratio is the ratio between the size of the compressed file and the size of the source file.

$$\text{Compression ratio} = \frac{size\ after\ compression}{size\ before\ compression}$$

Table-1 Compression ratio between existing and proposed algorithm

Algorithm type	**Compression ratio**
LZW	0.67
Huffman encoding	0.32
Reference based compression	0.59
BAR algorithm	0.86
Proposed Hybrid data compression algorithm	0.932

(b) Compression time:

Table-2 Compression time between existing and proposed algorithm

Algorithm type	**Average compression time for 100 kb**
LZW	0.25 sec
Huffman encoding	0.54 sec
Reference based compression	0.19 sec
BAR algorithm	0.43 sec
Proposed Hybrid data compression algorithm	0.152 sec

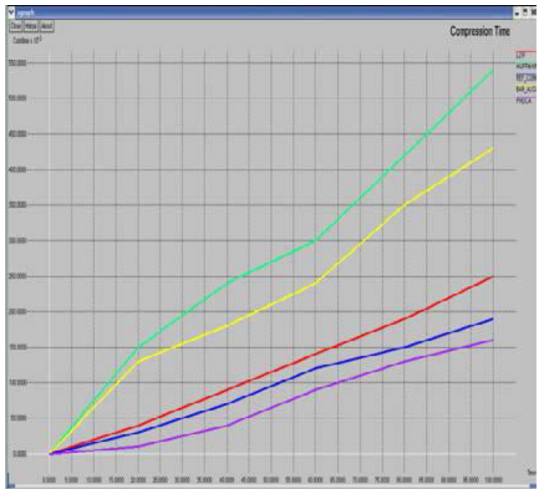

Figure 1 -Graphical representation of compression time

4. Conclusion

This paper work studies the security issues of ensuring the integrity of data storage in cloud computing. As the data get compressed, it leads to a more optimized way of retrieving data from cloud. The use of compression in cloud computing leads to effective use of storage disks and bandwidth. As a result the proposed algorithm achieves the compression time 0.16 sec and decompression time 0.21sec for 100 kb file.

References:

[1] C.C. Tan, Q. Liu, and J. Wu. Secure locking for untrusted clouds. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 131–138. IEEE, 2011.

[2] Qian Wang, Cong Wang, Jin Li, Kui Ren, and Wenjing Lou," Enabling Public Audit ability and Data Dynamics for Storage Security in Cloud Computing", IEEE computer society , Vol 22, No 5, May 2011

[3] Dr. Anil G.N, Mrs. Swetha M.S & Mr. Muneshwara M.S "A Smarter Way of Securing and Managing Data for Cloud Storage Applications Using High Throughput Compression in the Cloud Environment" IJARCSMS Volume 2, Issue 9, September 2014.
[4] "PPM performance with BWT Complexity: A fast and effective data compression algorithm", M. Effros,Proceedings of the IEEE, 88(11), 1703-1712, (2000).
[5] Yuan, Jiawei, and Shucheng Yu. "Secure and constant cost public cloud storage auditing with deduplication."