DATA INTEGRITY PROOFS IN CLOUD STORAGE

Bhushan Koli, Rahul Kute, Deepak Oza, Jigar Prajapati, Sachin Gavhane
Department of Information Technology
Atharva College of Engineering, University of Mumbai, India.
bhushank008@gmail.com, rbk3192@gmail.com, alkaoza44@yahoo.com, jigarprajapati15@yahoo.com
sachin2006g@yahoo.co.in

***Abstract* - To ensure data integrity, the endorsement copy of a file must be an accurate image of the unique at a given point in instant. However, replication of a file is not an direct process. Unless the file is very small, the support user must read from the file and write to the support medium numerous times to make a absolute copy. If the support user cannot make sure that no other application modifies the file while it is being unoriginal, you may have a problem with the integrity of the data being unoriginal. Since the high rate of data storage space devices and the rapid expansion of data generated is very costly for enterprises or users to regularly update their devices. In cloud storage, data is being transferred to data centers which are distantly situated and users don't have right to use these data centers.**

There should be a method for the user to check if the integrity of the data is maintained or is compromised. In this paper we offer a idea which helps the users to verify the rightness of data in the cloud. This evidence is approved upon by both the cloud and the user and can be included in the Service Level Agreement (SLA). With this idea the user side has to maintain minimal storage which is advantageous for the user.

Keywords – **Service Level Agreement (SLA), Proof of Retrievability, Data Integrity.**

I. INTRODUCTION

Since Data Outsourcing is of financial raising it is a growing fashion among the cloud data storage. This basically means that the user of the data moves its data to a third party cloud storage server which authentically stores the data with it and provide it back to the owner whenever necessary.

Whenever extra data is created it becomes complicated for small firms to inform their hardware and also maintaining the storages can be a complicated task. Small firms decrease their storage price by outsourcing their data to cloud storage. The cloud storage also maintain several copies of user's data thereby falling the chance of losing data by hardware crash.

In this paper we contract with the difficulty of implementing a protocol for obtaining a evidence of data control in the cloud sometimes referred to as Proof of Retrievability (POR). This idea tries to gain a proof to facilitate the information being stored by the user at a distant data storage in the cloud is not modified by anyone. This helps a system to stay away from the cloud storage from misinterpreting and modifying the data stored at it without the consent of the data owner by using frequent checks on the storage archives. While storing data at the cloud storage space we are restricted by the resources at the cloud storage servers. Since the data volume of the user is very huge and stored at distant places, accessing the complete file can be costly in I/O costs to the storage server and also transmitting the complete file across network consumes a large bandwidth. Since the increase in storage capacity has far out spaced the increase in data right to use as well as network bandwidth, accessing and transmitting the complete records even infrequently greatly restrictions the scalability of the network resources. This problem is further more difficult by the fact that the owner of the data may be a small device, like PDA (Personal Digital Assist) or a cell phone, which have less CPU power, battery power and communication bandwidth. So the idea needs to be able to produce a proof without the need of the server to right to use the complete file or the client retrieving the complete file from the server. Also the idea should reduce the local working out at the client as well as the bandwidth consumed at the client.

II. PROBLEM DEFINITION

Storing of user data in the cloud even though its advantages has many interesting security concerns which need to be widely investigated for making it a consistent solution to the problem of avoiding local storage of data. Many problems

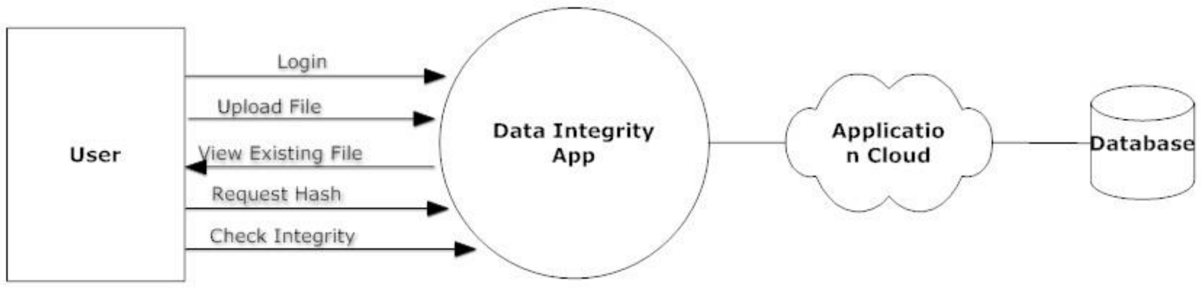

Fig. 1 Architecture

like data verification and integrity (i.e., how to professionally and steadily ensure that the cloud storage server returns accurate and entire results in response to its clients' queries, outsourcing encrypted data and related complicated problems dealing with querying over encrypted area were discussed in study literature.

III. PROPOSED SYSYTEM

Synchronization of files such as pictures, document, sound etc. The user interface will show a multiple option to arrange, organize etc to the main device. The availability of following options are , the data files to be arranged in order required ,duplicate files to be removed, files to be stored in other device. One of the most important concerns that need to be addressed is to assure the customer of integrity i.e. the correctness of his data in the cloud. In this paper we offer a idea which gives us a proof of data integrity in the cloud which the customer can employ to check the correctness of his data in the cloud. Service level agreement (SLA) this proof can be agreed by both cloud and customer. It is important to note that our proof of data integrity protocol just checks the integrity of data. Any file, media or document added to the destination. The user open the same folder on the other device the added file will be present there will be present there with the help of synchronization process. There will be common server for one working environment. The multiple clients will be provided with a server which will work as cloud storage. For one company, one organization or can be for one branch!

IV. METHODOLOGY

Message Digest 5 (MD5) Algorithm

MD5 is a message by digest algorithm developed Ron Rivest at MIT.

The algorithm takes as input a message-bit of arbitrary length and produces as output a 128 message digest of the input.

This is mostly intended for digital signature applications where a large file must be compressed in a secure manner before being encrypted with a private (secret) key under a public key cryptosystem.

Steps to calculate Message Digest of the input message:

Step 1: Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is equal to a length which is 64-bits less than exact multiple of 512. That is, the message is extended so that it is just 64 bits shy of being a multiple of 512 bits long.

Padding should be performed even if the length of the original message is equal to 64-bits less than exact multiple of 512.

Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes 64-bits less than exact multiple of 512-bits.

Step 2: Append Length

After padding bits are added, next step is to calculate the original length of the message and add it to the end of the message, after padding. The length of the message is calculated, excluding the padding bits (i.e. it is the length before the padding bits were added).

This length of the original message is now expressed as a 64-bit value and these 64 bits are appended to the end of the original message + padding. If the length of the message exceeds 2^{64} bits, then the lower-order 64 bits of the length are used.

Step 3: Divide the input into 512-bit block

Now divide the input message into blocks, each of length 512 bits.

Step 4: Initialize Chaining Variables

A four-word buffer (A, B, C, and D) is used to compute the message digest.

Here each of A, B, C, D is a 32-bit register which are initialized to the following values in hexadecimal.

Word A: 01 23 45 67
Word B: 89 ab cd ef
Word C: fe dc ba 98
Word D: 76 54 32 10

Step 5: Process Message

Step 5.1: Copy the four variables into four corresponding variables, a, b, c and d. Thus, we have a=A, b=B, c=C and d=D.

Step 5.2: Divide the current 512-bit block into 16 sub-blocks i.e. each sub-block is of size 32 bits.

Step 5.3: Now, there are four rounds. In each rounds, all the 16 sub-blocks are processed belonging to a block. The input to each round are: (a) all the 16 sub-blocks, (b) the variables a, b, c, d and (c) some constant, t.

All the four rounds vary in one major way: Step 1 of the four rounds has dissimilar processing. The further steps in all the four rounds are the similar.

- In each round, there are 16 input sub-blocks, M[0], M[1], ….., M[15] or in general, M[r], where R varies from 0 to 15.
- Also t is any constants array which contains 64 elements, with each element consisting of 32 bits, they are denoted as t[1], t[2],….. t[64] or in general, t[k], where k can take values from 1 to 64. Each the four rounds takes 16 values of the 64 values of t.

The iterations of the all four rounds is as follows:

A process P is Performed on b, c and d. This process is different in all the four rounds.

- The variable a is added to the output of the process p (i.e. to register abcd).
- The message sub-block M[r] is added to the output of Step 2.
- To the output of Step 3, t[k] which is a constant is added.
- The output of Step 4 is circular-left shifted by 's' bits (the value of s keeps changing).
- To the output of Step 5, variable b is added.
- The value of abcd obtained in Step 6 now becomes the abcd value for next round.

Process P for each of the four rounds:
Round 1: (b AND c) OR ((NOT b) AND (d))
Round 2: (b AND d) OR (c AND (NOT d))
Round 3: b XOR c XOR d
Round 4: c XOR (b OR (NOT d)).

V. CONCLUSION

In this paper we proposed MD5 algorithm which gives the user a proof of data integrity. It provides flexible and cost effective alternatives to store data at cloud. This algorithm can be used to construct a constant length message digest which is 128-bits irrespective of the size of the message.

REFERENCES

[1] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrityin outsourced databases," Trans. Storage, vol. 2, no. 2, pp. 107–138, 2006.

[2] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searcheson encrypted data," in SP '00: Proceedings of the 2000 IEEE Symposiumon Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2000, p. 44.

[3] A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for largefiles," in CCS '07: Proceedings of the 14th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2007, pp. 584–597.

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in CCS '07: Proceedings of the 14th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2007, pp. 598–609.

[5]. R. Rivest, The MD5 Message-Digest Algorithm, RFC 1321, MIT LCS & RSA Data Security, Inc., April 1992.

[6]Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport :Baliga, J.; Ayre, R.W.A.; Hinton, K.; Tucker, R.S.; Proceedings of the IEEE Volume: 99 , Issue: 1 Digital Object Identifier: 10.1109/JPROC.2010.2060451 Publication Year: 2011 , Page(s): 149 - 167 Cited by: 1.

[7]Analysis and Research of Cloud Computing System Instance: Shufen Zhang; Shuai Zhang; Xuebin Chen; Shangzhuo Wu; Future Networks, 2010. ICFN '10. Second International Conference on Digital Object Identifier: 10.1109/ICFN.2010.60 Publication Year: 2010 , Page(s): 88 – 92.