Data Deduplication Scheme for Cloud Storage

[1]Iuon-Chang Lin and [2]Po-Ching Chien

Abstract

Nowadays, the utilization of storage capacity becomes an important issue in cloud storage. In this paper, we introduce two categories of data deduplication strategy, and extend the fault-tolerant digital signature scheme proposed by Zhang on examining redundancy of blocks to achieve the data deduplication. The proposed scheme in this paper not only reduces the cloud storage capacity, but also improves the speed of data deduplication. Furthermore, the signature is computed for every uploaded file for verifying the integrity of files.

1. Introduction

Owing to the population of cloud service and the increase of data volume, more and more people pay attention to economize the capacity of cloud storage than before. Therefore, how to utilize the cloud storage capacity well becomes important issue nowadays. Data deduplication is a technique to identify those data which have the same contents and only store one copy of them. Therefore, data deduplication can economize the cloud storage capacity and utilize cloud storage more properly.

According to the original cloud storage schemes, some of schemes store the whole file into the storage server without any deduplication. Thus, if there are two similar files, the cloud storage server would store redundant blocks between these two similar files. Therefore, the cloud storage capacity cannot be used properly. There are some cloud storage vendors using the technique of data deduplication when storing the uploaded files, the DropBox for example. Some data deduplication schemes calculate a hash value for each file used to check whether there is redundant hash value among uploaded files in the cloud storage. Others translate a file into n blocks and then calculate a hash value to represent every block; therefore, the cloud storage server can examine the redundancy of every hash value of blocks.

Though this method can find those blocks not stored in the cloud storage server, it spends too much time on examining these duplicate blocks. In this paper, the proposed scheme not only can utilize the capacity of cloud storage server more properly, but also improve the speed of data deduplication. Furthermore, a signature is computed for every uploaded file to ensure the integrity of file.

2. Related Work

In this section, the categories of data deduplication strategy and the Zhang’s fault-tolerant digital signature scheme which is the basic scheme of the proposed scheme is introduced.

[1]*Dept. of Information Management National Chung Hsing University, and Department of Photonics and Communication Engineering, Asia University*
Email: iclin@nchu.edu.tw
[2]*Dept. of Information Management National Chung Hsing University, Taichung, Taiwan*
Email: becky6110@gmail.com

2.1 Categories of Data Deduplication Strategy

Deduplication strategy can be categorized into two main strategies as follow, differentiated by the type of basic data units.

1).File-level deduplication: A file is a data unit when examining the data of duplication, and it typically uses the hash value of the file as its identifier. If two or more files have the same hash value, they are assumed to have the same contents and only one of these files will be stored.

2).Block-level deduplication: This strategy segments a file into several fixed-sized blocks or variable-sized blocks, and computes hash value for each block for examining the duplication blocks.

2.2 View of Zhang's Fault-Tolerant Digital Signature Scheme

Zhang's digital signature scheme with fault tolerance is based on RSA cryptosystem. A user has a RSA modulus *N*, *N*= $p \times q$, which *p* and *q* are two large primes. Then the user chooses a public key *e* randomly, and secret key *d*, *e* and *d* must satisfy the equation $d = e^{-1} \bmod (p-1)(q-1)$. Let (e_A, N_A) be the public keys of user A, (e_B, N_B) be the public keys of user B. And d_A and d_B be the private key of user A and B respectively. Besides, assume $N_A \neq N_B$ and the length of N_A and N_B is the same of simplification.

Step 1: User B translates the message M into an n×m matrix and sends the matrix to user A.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}$$

where x_{ij}, 1≤i≤n, 1≤j≤m, is a message block which has the same length as N_A and N_B.

Step 2: For the given n×m message matrix X, B constructs an (n+1)×(m+1) matrix X_h as follow:

$$X_h = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} & X_1 \\ x_{21} & \cdots & \cdots & x_{2m} & X_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} & X_n \\ \widetilde{X}_1 & \widetilde{X}_2 & \cdots & \widetilde{X}_m & h \end{bmatrix}$$

where the parameters X_i,$\widetilde{X}_j$ andh:

$$X_i = \prod_{j=1}^{m} x_{ij} \bmod N_B \text{ for } 1 \le i \le n, \qquad (1)$$

$$\widetilde{X}_j = \prod_{i=1}^{n} x_{ij} \bmod N_B \text{ for } 1 \le j \le m, \qquad (2)$$

and

$$h = \prod_{j=1}^{m}\left(\prod_{i=1}^{n} x_{ij} \bmod N_B\right) \bmod N_B. \qquad (3)$$

Step 3: B computes an (n+1)×(m+1) ciphered matrix C_h as follow:

$$C_h = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} & C_1 \\ c_{21} & \cdots & \cdots & c_{2m} & C_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} & C_n \\ \widetilde{C}_1 & \widetilde{C}_2 & \cdots & \widetilde{C}_m & h_c \end{bmatrix}$$

where the parameters c_{ij}, C_i,$\widetilde{C}_j$and h_c:

$$c_{ij} = x_{ij}^{e_A} \bmod N_A, \qquad (4)$$

$$C_i = X_i^{e_A} \ mod \ N_A, \quad (5)$$

$$\widetilde{C_j} = \widetilde{X_j}^{e_A} \ mod \ N_A, \text{and} \quad (6)$$

$$h_c = h^{d_B} \ mod \ N_B, \text{ for all } 1 \le i \le n \text{ and } 1 \le j \le m. \quad (7)$$

Step 4: After receiving C_h, A decrypts C_h for obtaining message X_h' by using his private key d_A. The decrypted message is as follow:

$$X_h' = \begin{bmatrix} x'_{11} & x'_{12} & \cdots & x'_{1m} & X'_1 \\ x'_{21} & \cdots & \cdots & x'_{2m} & X'_2 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x'_{n1} & x'_{n2} & \cdots & x'_{nm} & X'_n \\ \widetilde{X'}_1 & \widetilde{X'}_2 & \cdots & \widetilde{X'}_m & h' \end{bmatrix}$$

where the parameters x'_{ij}, X'_i, $\widetilde{X'}_j$ and h':

$$x'_{ij} = c_{ij}^{d_A} \ mod \ N_A, \quad (8)$$

$$X'_i = C_i^{d_A} \ mod \ N_A, \quad (9)$$

$$\widetilde{X'_j} = \widetilde{C_j}^{d_A} \ mod \ N_A, \text{and} \quad (10)$$

$$h' = h_c^{e_B} \ mod \ N_B. \quad (11)$$

Step 5: A checks whether $X'_i = \prod_{j=1}^{m} x'_{ij} \ mod \ N_B$, $X'_j = \prod_{j=1}^{n} x'_{ij} \ mod \ N_B$, and $h' = (X'_1 \times X'_2 \times \cdots \times X'_n) = (\widetilde{X'}_1 \times \widetilde{X'}_2 \times \cdots \times \widetilde{X'}_m)$ or not. If the verifications are positive, user A can believe that the message is indeed sent by user B. Otherwise, there are some errors in the decrypted message.

Step 6: Then user A can detect the error by the following two equations:

$$x'_k = \prod_{j=1}^{m} x'_{kj} \ mod \ N_B, \text{ and} \quad (12)$$

$$x'_l = \prod_{i=1}^{n} x'_{il} \ mod \ N_B. \quad (13)$$

Assume that the error occurs in the message block x_{kl}.

Step 7: User A can correct the error by computing one of the following equations:

$$x'_{kl} = X'_k \times \left(\prod_{j=1, j \ne l}^{m} x'_{kj}\right)^{-1} mod \ N_B, \quad (14)$$

$$x'_{kl} = X'_l \times \left(\prod_{i=1, i \ne k}^{n} x'_{il}\right)^{-1} mod \ N_B. \quad (15)$$

3. Our Proposed Scheme

The scheme proposed by Zhang can detect and correct errors efficiently in digital signature. Based on top of Zhang's scheme, this paper proposes a novel data deduplication method to improve the utilization of cloud storage capacity and the speed of deduplication in cloud storage.

The file is translated into (n*m) blocks, and h_{ci} and $\widetilde{h_{c1}}$ is computed as the feature values to represent every row and column. The proposed scheme is separated into two phases. First phase is file translation, and the second phase is data deduplication examination in cloud storage.

3.1 File Translation Phase

Suppose that user A uploads the file C to a cloud storage,

Step 1: The cloud storage translates C into (n×m) blocks, builds a (n+1)×(m+1) matrix to store (n × m) blocks, and computes parameter h_{ci} for each row, parameter $\widetilde{h_{cj}}$ for each column, and parameter Z_c as follow:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} & h_{c1} \\ c_{21} & \cdots & \cdots & c_{2m} & h_{c2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} & h_{cn} \\ \widetilde{h_{c1}} & \widetilde{h_{c2}} & \cdots & \widetilde{h_{cm}} & Z_c \end{bmatrix}$$

where the parameters h_{c1}, $\widetilde{h_{c1}}$ and Z_c:

$$h_{ci} = \prod_{j=1}^{m} H_1^j(c_{ij}) \bmod N_A \ for\ 1 \le i \le n, \tag{16}$$

$$\widetilde{h_{cj}} = \prod_{i=1}^{n} H_2^i(c_{ij}) \bmod N_A \ for\ 1 \le j \le m, \text{and} \tag{17}$$

$$Z_c = H_3[\prod_{i=1}^{n}(h_{ci}) \prod_{j=1}^{m}(\widetilde{h_{cj}})]. \tag{18}$$

Step 2: After user A receives Z_c, S_c, as the signature of file C is calculated by using his private key d_A and N_A which is the block number of file C. Therefore, user A can confirm the integrity of file C when downloading this file in the future.

$$S_c = (Z_c)^{d_A} \bmod N_A \tag{19}$$

Step 3: When the cloud storage server receives S_c, the parameter Z_c in the (n+1)×(m+1) matrix will be replaced as S_c as follow.

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1m} & h_{c1} \\ c_{21} & \cdots & \cdots & c_{2m} & h_{c2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{n1} & c_{n2} & \cdots & c_{nm} & h_{cn} \\ \widetilde{h_{c1}} & \widetilde{h_{c2}} & \cdots & \widetilde{h_{cm}} & S_c \end{bmatrix}$$

3.2 Data Deduplication Phase

After user A uploads another file C´, the cloud storage server translates C´ into a (n+1)×(m+1) matrix as follow in the first phase, and the cloud storage server starts to examine the redundancy of C´.

$$C' = \begin{bmatrix} c'_{11} & c'_{12} & \cdots & c'_{1m} & h_{c'1} \\ c'_{21} & \cdots & \cdots & c'_{2m} & h_{c'2} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c'_{n1} & c'_{n2} & \cdots & c'_{nm} & h_{c'n} \\ \widetilde{h_{c'1}} & \widetilde{h_{c'2}} & \cdots & \widetilde{h_{c'm}} & S_{c'} \end{bmatrix}$$

where the parameters $h_{c'i}$, $\widetilde{h_{c'j}}$ and $S_{c'}$:

$$h_{c'i} = \prod_{j=1}^{m} H_1^j(c'_{ij}) \bmod N_A \ for\ 1 \le i \le n, \tag{20}$$

$$\widetilde{h_{c'j}} = \prod_{i=1}^{n} H_2^i(c'_{ij}) \bmod N_A \ for\ 1 \le j \le m, \tag{21}$$

$$Z_{c'} = H_3[\prod_{i=1}^{n}(h_{c'i}) \prod_{j=1}^{m}(h_{c'j})], \text{and} \tag{22}$$

$$S_{c'} = (Z_c)^{d_A} \bmod N_B. \tag{23}$$

Cloud storage server examines the redundancy by checking whether

$$h_{c'i} = h_{ci} \text{ and} \tag{24}$$

$$\widetilde{h_{c'j}} = \widetilde{h_{cj}}. \tag{25}$$

If the examination results are positive, it means file C´ has the same contents as file C. If the examination results are shown as follow:

$$h_{c'i} \neq h_{ci} \text{and} \tag{26}$$

$$\widetilde{h_{c'j}} \neq \widetilde{h_{cj}}, \tag{27}$$

it confirms that the block c'_{ij} differs from block c_{ij}.

After the cloud storage server verifies every corresponding $h_{c'i}$ and $\widetilde{h_{c'j}}$, the redundancy examination phase is completed. The cloud storage server will store these in different blocks, not being stored in the cloud storage server previously. These different blocks are named as new blocks, and the same blocks as old blocks respectively.

4. Discussions

4.1 Worst Case Conditions

Different position of new blocks can result in different examination conditions. In the proposed scheme, the worst case refers that the matrix has only max(n,m) new blocks, but the cloud storage server regards all blocks as new blocks in the data deduplication phase, and thus all blocks are stored into the cloud storage server.

Assume that there is a (n×m) matrix, and n>m. When there are n new blocks in a matrix, and m new blocks of these new blocks are distributed to every different row and column in the matrix, then the worst case appears. Figure 1 is an example of the worst case condition. The notation × in Figure 1 means the new blocks, and the natation ○ in Figure 1 means the old blocks.

$$\begin{bmatrix} \times & \circ & \circ & \circ \\ \circ & \times & \circ & \circ \\ \circ & \circ & \times & \circ \\ \circ & \circ & \circ & \times \\ \circ & \times & \circ & \circ \end{bmatrix} \quad \begin{bmatrix} \circ & \circ & \circ & \circ & \times \\ \circ & \times & \circ & \circ & \circ \\ \times & \circ & \circ & \times & \circ \\ \circ & \circ & \times & \circ & \circ \end{bmatrix} \quad \begin{bmatrix} \circ & \circ & \circ & \times \\ \circ & \circ & \times & \circ \\ \circ & \times & \circ & \circ \\ \times & \circ & \circ & \circ \end{bmatrix}$$

Figure 1: Three examples of the worst case condition.

After the worst case condition is discovered, the probability of appearance of the worst case can be formulated as

$$\frac{C_m^n \times m!}{(n \times m)!} \qquad (28)$$

$(n \times m)!$: all combinations of new block in the matrix.

C_m^n: select m non-redundant rows or columns from n rows or columns as new blocks.

$m!$: the combinations of new blocks in the selected rows or columns.

Assume there is a 4×5 matrix, n=5 and m=4. The probability of appearance of the worst case is $4.93238115\times 10^{-17}$. This indicates that the probability of appearance of the worst case is relatively low.

4.2 Computation Cost

The data deduplication strategy used in proposed scheme is the fixed-sized blocks of block-level deduplication; therefore, the comparison of the proposed scheme with Venti archival storage system is stated in Table 1. The computation cost indicates the complexity of translating a file into a matrix. The frequency of block verification represents the complexity of verifying the hash value between uploaded file blocks and cloud storage blocks to check whether the same blocks exist in the cloud storage or not.

From Table 1 it is clear that our proposed scheme uses less time on block verification than Venti's. The reason is the proposed scheme checks only one hash value for each column and row instead of verifying the hash value of every block.

Table 1: The computation cost and frequency of block verification on data deduplication phase of our proposed scheme and Venti archival storage system

	Proposed scheme	Venti
Computation Cost	hash: O(n^2) multiple: O(n)	hash: O(n^2)
Frequency of Block Verification	O(n)	O(n^2)

5. Conclusions

To be concluded, the proposed scheme not only enhances the efficiency of data deduplication, but also improves the speed of data deduplication phase by calculating a feature value to represent every column and row instead of verifying the hash value of every block. Though there is a problem of the worst case in the proposed scheme that the cloud storage server will regard all blocks as new blocks and store all of these blocks, resulting in storing duplicate blocks, the probability of the worst case is low and won’t affect most.

References

[1] D.Harnik, B. Pinkas, and A. Shulman-Peleg,“Side Channels in Cloud Services: Deduplication in Cloud Storage,” IEEE Security and Privacy, vol.8, no.6, pp.40-47,2010.

[2] W. K. Ng, Y. Wen, and H. Zhu, “Private Data Deduplication Protocols in Cloud Storage,” Proc. 27th Ann. ACM Symposium on Applied Computing, pp. 441-446, 2012.

[3] H.S. Gunawi , N. Agrawal , A.C. Arpaci-Dusseau , R.H. Arpaci-Dusseau , and J. Schindler, “Deconstructing Commodity Storage Clusters,” Proc. 32nd Ann. Int’l Symp. Computer Architecture (ISCA 05), IEEE CS Press, pp. 60–71, 2005.

[4] John R. Douceur , A.Adya , W.J. Bolosky , D. Simon , and M. Theimer,, “Reclaiming Space from Duplicate Files in a Serverless Distributed File System,” Proc. Int’l Conf. Distributed Computing Systems, IEEE CS Press, pp. 617–630, 2002.

[5] S. Quinlan and S. Dorward, “Venti: A New Approach to Archival Storage,” Proc. 1st Usenix Conf. File and Storage Technologies (FAST ’02), Usenix Assoc., pp. 89–101, 2002.

[6] A. Muthitacharoen, B. Chen, and D. Mazieres, “A Low-Bandwidth Network File System,” Proc. Symp. Operating Systems Principles (SOSP 01), ACM Press, pp. 174–187, 2001.

[7] L.L. You, K.T. Pollack, and D.D.E. Long, “Deep Store: An Archival Storage System Architecture,” Proc. 21st Int’l Conf. Data Eng. (ICDE 05), IEEE Press, pp. 804–815, 2005.

[8] C.N. Zhang, “Integrated Approach for Fault Tolerance and Digital Signature in RSA, ”IEE Proc. Computers & Digital Techniques, vol. 146, no. 3, pp. 151–159, 1999.

[9] I.C. Lin and C.C. Chang, “Security Enhancement for Digital Signature Schemes with Fault Tolerance in RSA,” Information Sciences, vol. 177, no. 19, pp. 4031–4039, 2007.

[10] I.C. Lin and C.C. Chang, “An Efficient Fault-Tolerant Digital Signature Scheme Based on the Discrete Logarithm Problem,” Autonomic and Trusted Computing, vol. 4158, pp. 601-610, 2006.