International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.4 (2016), pp. 9-16
http://dx.doi.org/10.14257/ijmue.2016.11.4.02

An Efficient Provable Data Possession Scheme based on Counting Bloom Filter for Dynamic Data in the Cloud Storage

Eunmi Jung[1] and Junho Jeong[2]

[1, 2] Department of Computer Engineering, Dongguk University
30, Pildong-ro 1 gil, Jung-gu, Seoul, Republic of Korea
{lovest0394[1],yanyenli[2]}@dongguk.edu

Abstract

Cloud services have become a trend, because many companies are supplying various cloud services that reflect the needs of users. However, as cloud services increase, security problems, such as saved data spills and data modulation, also increase. This paper focuses on the problem of data integrity. A PDP (Provable Data Possession) scheme has the disadvantage of time overheads, because dynamic data needs additional processing time. To solve the problem, this paper examines the use of a Counting Bloom Filter for updating data. This technique is more efficient than using PDP when changing part of the data in a simulation. The result is more effective if the saved data changes part of the dynamic data.

Keywords*: cloud computing security, data integrity, provable data possession*

1. Introduction

As the popularity of the cloud increases, more people are using various services based on the cloud. In recent years, the cloud has been expanded to include additional services, such as providing service platforms. As a cloud environment is popular, high accessibility and high-capacity storage have become issues. When users save data to cloud storage, the saved data can be accessed from everywhere without having to go through a USB and external hard drive. In addition, low cost for the amount of storage space provided is an important point for selecting the cloud environment. Due to the affordable cost of using cloud storage, cloud storage has been used to build servers and databases in many companies [1].

When users upload data, the data is stored in cloud storage. A public cloud can be accessed by other users, because the cloud does not provide available storage space as a personal storage environment. This causes problems, such as a malicious user accessing and abusing another user's data for their own benefit [2-4]. This decreases the reliability of the cloud service providers, which would eventually lead to fewer cloud users. Thus, cloud security issues are very important [5]. Therefore, there is a need for ways to address the security of data stored in the cloud. One method is to verify the integrity of stored data. The basic method of data integrity verification is an exhaustive search method to verify whether or not through the control of the entire data modulation. Various schemes for verifying data integrity are currently being studied.

2. Related Works

2.1. Provable Data Possession

PDP is a data integrity verification method [6]. Among other integrity verification techniques are the techniques of PORs [7]. A PDP scheme consists of two phases: a pre-process and the verification process. Pre-process creates and stores metadata for saved

ISSN: 1975-0080 IJMUE

data. The verification process of PDP is via the metadata generated by the pre-process, query and response statement. In the verification process, the user requests information about a part of the file to the server. Figures 1 and 2 show the two phases of PDP [6].

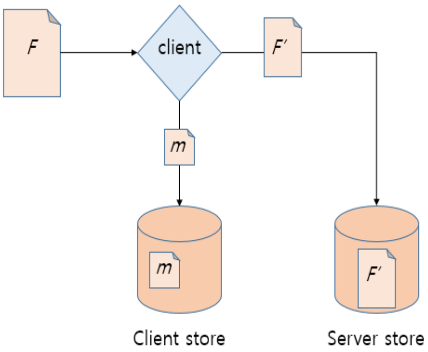

Figure 1. Pre-Process in PDP

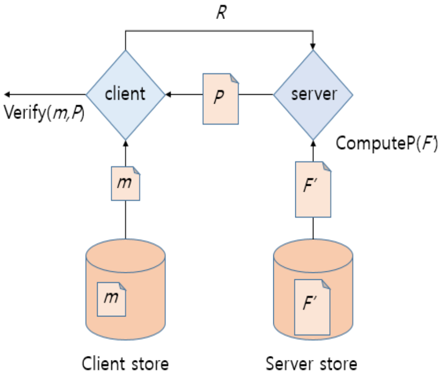

Figure 2. Verification-Process in PDP

A cloud storage server accesses the distributed files to obtain part of the file. The advantage of PDP is to determine whether there has been a forgery or the data has only partial information without using an exhaustive search method. Since the user does not need to perform the process of downloading the whole data for an exhaustive search verification, which makes the process more cost-effective. Data integrity verification schemes based on PDP have been studied. Increasing the effect of the PDP has two elements.

First, advanced schemes consider the size of the metadata and query statements. If the size difference between query statement and entire file is little, the PDP advantage is low. Furthermore, the size of the query statement is associated with the metadata.

Second, advanced schemes should consider the computation cost of the calculations operated in the pre-process to generate the metadata and the verification process to generate the query statements and responses. If the efficiency of the PDP is reduced, due to the complexity in carrying out the operation is required to have high operation costs calculated for each course.

Y. Zhu *et al.* proposed a new PDP Cooperative Provable Data Possession (CPDP) [8]. CPDP is a method applied to the PDP using the bilinear group, hash function and aggregation algorithm.

Method-based CPDP have been studied. The PDP scheme proposed by X. Sun *et al.* [9] is one method in which the PDP has to recreate the metadata when the new data is registered or changed. This paper proposed a scheme capable of partly inserting additional data into the data to change. That method is possible to partial insertion of the data. So, the proposed scheme is to not recreate new metadata for the modified data. A PDP scheme that can reduce communication costs is also being studied.

Purshothama B.R. *et al.* proposed a method [10] using a bilinear map. In this paper, the server and the user each generate a key, and exchange their keys, thereby generating a tag for each block divided by the corresponding key. This is similar to the scheme of CPDP, because it has a way to define and store the metadata. In the verification process, the user transmits the index of the data to be verified, and the server generates the verification data to access the data for the index. Depending on the nature of the bilinear map, the verification information has the effect of reducing communication costs due to the transmission of compressed information.

T. Shuang Gibbon *et al.* proposed a scheme that combines the Data Signature in a PDP [11]. In this paper, they use the homomorphic features of the RSA algorithm. Operations between the cipher-text results in homomorphic features means that the cryptography is equal to the result obtained by encrypting the original text results of the cipher-text. Thus, operation is possible between the states in the encrypted data without the decoding process [12]. In this paper, creating a digital signature on a piece of divided data and using homomorphic characteristics by calculating the number of signatures into a single signature is used for integrity verification. When creating a query, the time savings and cost reduction is possible due to the transmission of a signature to be verified. If the data update occurs frequently, the PDP is to repeat the preprocessing to update the metadata. If the number of times to change the file is ever an issue but more update count increases, performance degradation occurs. PDP techniques for improving the performance of the dynamic data is also being studied.

C. Li *et al.* proposed PDP techniques to insert and edit a new data block to the stored data [13]. This generates a table based on the index of the location index and the physical location of the logical block stored in the block. PDP verification is possible because the data tag can be verified with this operation for the table. Even if the data is modified, the proposed scheme has the effect of reducing computational costs. Thus, the study of PDP for dynamic data is required to perform integrity verification for the data to be updated.

2.2. Counting Bloom Filter

The Bloom Filter proposed by B. Bloom is used primarily to determine whether or not to include any probabilistic data structure elements with respect to the set [14]. It uses less memory space since the Bloom Filter is composed of a bit array, and the operation is also faster. The insertion operation of the Bloom Filter can obtain the index value of the data through the hash function, and sets the bit in the index. However, the disadvantages of the Bloom Filter can result in a false-positive rate, because of the nature of the hash, and the fact that the delete operation is not possible. Bloom Filter is also studied techniques to ensure integrity in cloud [15].

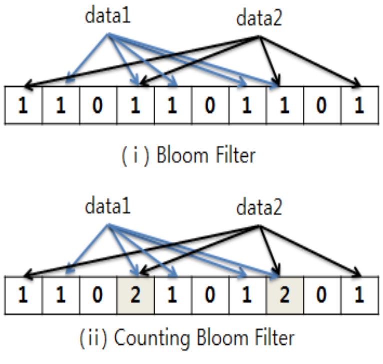

Figure 3. Bloom Filter and Counting Bloom Filter

The Counting Bloom Filter is a mechanism that improves the Bloom Filter to allow the delete operation [16]. With the delete operation, the Counting Bloom Filter can change the value of the index. Figure 3 shows a Bloom Filter and Counting Bloom Filter's insertion operation for test data. The Counting Bloom Filter increases the index when the data has the same index information, but the Bloom Filter has only 1 bit of data for the same test data. To take advantage of Counting using buckets, if applied to a Counting Bloom Filter to the PDP, will be more effective for updating the data frequently when performing iterative operations.

3. Provable Data Possession Using the CBF

The idea of this paper is to apply the Counting Bloom filter to a Provable Data Possession scheme. Like previous the PDP using the Bloom Filter [17], the Counting Bloom filter PDP scheme in this paper is applied to create a query in the verification process and metadata. Since the Counting Bloom Filter is able to delete part of the indexed blocks, the metadata using can be changed.

This technique is effective when the stored data is frequently updated. The PDP scheme does not provide an update process for stored data, which causes some performance degradation, because of the increase in the overhead, as the pre-processing operation is carried out for a typical data redundancy. The PDP scheme adds a data update using a counting Bloom filter is shown in Figure 4.

Figure 4 shows that the user can modify the blocks in the stored data. In our proposed scheme, the metadata can be corrected through the Update_metadata function. After the update-process, the user transfers the modified or revised files to the server. For the Hadoop system, even if some files are modified, it is not possible to modify some of the information that should be transmitted to the entire modified file.

The PDP scheme performs the pre-process and verification process, but our proposed scheme has an additional update process. During the phase of updating data via a counting Bloom filter, the scheme may generate metadata without repeating the pre-process. If users need to update stored file, processing have to modify the corresponding block in the cloud and re-calculate hash values for the block set in the metadata. The Counting Bloom Filter may delete the information of the previous block using a delete operation. Therefore, this technique does not perform an additional operation, since replacing more updated block information to generate a bloom filter for the entire file to the metadata update block will be an effective way.

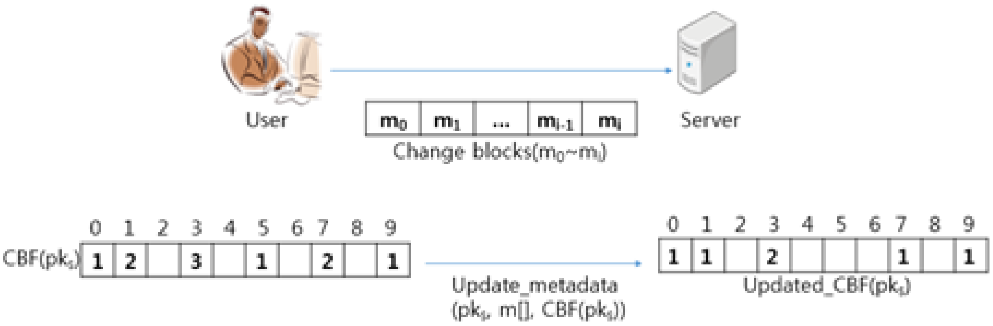
User
Server
m_0 m_1 ... m_{i-1} m_i
Change blocks(m_0~m_i)
0 1 2 3 4 5 6 7 8 9
CBF(pk_s) 1 2 _ 3 _ 1 _ 2 _ 1
Update_metadata
(pk_s, m[], CBF(pk_s))
0 1 2 3 4 5 6 7 8 9
1 1 _ 2 _ _ _ 1 _ 1
Updated_CBF(pk_s)

Figure 4. Update Process for Modified Blocks in CBF-PDP: The Update Function Performs the Role of Modifying the Currently Stored Metadata. The Value of the Index Block is Deleted by Using the Deletion Operation of the Counting Bloom Filter, and the Index of the New Modified Block, m_i

4. Simulation

While we measured the performance for our proposed scheme, we performed two experiments in a cloud environment. First, we executed a process to compare the performance of the Bloom filter, which was measured, to that of a counting Bloom filter (4.2). The second implementation of the proposed method was to measure the performance on an actual cloud (4.3).

4.1. Environment

The experiment configured the cloud environment to measure the performance of the technique to verify the integrity in the cloud. The cloud used in this simulation consisted of the Name and data nodes. The Name nodes are connected to the storage of data nodes that store the actual data [18-19]. In this paper, we configured the cloud environment with 12 data nodes. We used a Hadoop project to simulate a cloud environment.

4.2. Compare BF and CBF

A performance comparison was made by uploading an arbitrary file to the cloud with each file by using Bloom and Counting Bloom Filters to measure how long it takes to create each filter.

Using filters to generate metadata showed similar results. Figure 5 shows the results of measuring the time of generating a filter for the test data. The generation time of the Counting Bloom and the Bloom filter takes about 23 seconds.

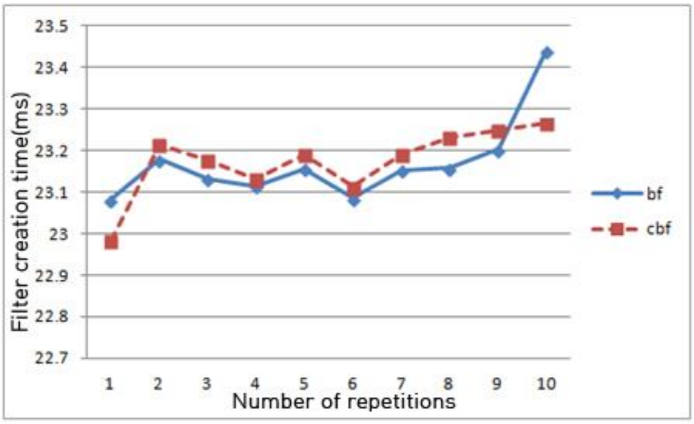

Figure 5. Creation Time of Filters

The update time is composed of the sum of the time to upload the time and new data to generate the metadata in the cloud. This ensures that the communication time is additionally required. The communication time is increased according to the increase in the size of the transmission file. This means that the communication time is proportional to the size of the file. Figure 6 shows the file transfer time for the block rates.

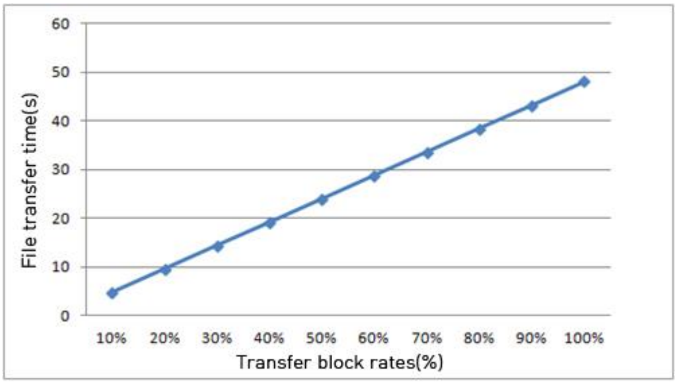

Figure 6. File Transfer Time for Block Rates

4.3. Simulation in HDFS

The data used in this simulation were used as a video file of 1.4 GB. The rate at which the files were updated and modified was in increments of 10% of the entire file. Simulations for the each condition of 100 repetitions were performed to measure the time that of the performance of the operations.

We tested the performance of the proposed scheme on the cloud environment by configuring the cloud environment using a HDFS (Hadoop Distributed File System) and performed the test. The operational point of view, the two filters have only difference on delete operation. Therefore, the time required to generate the filter is similar.

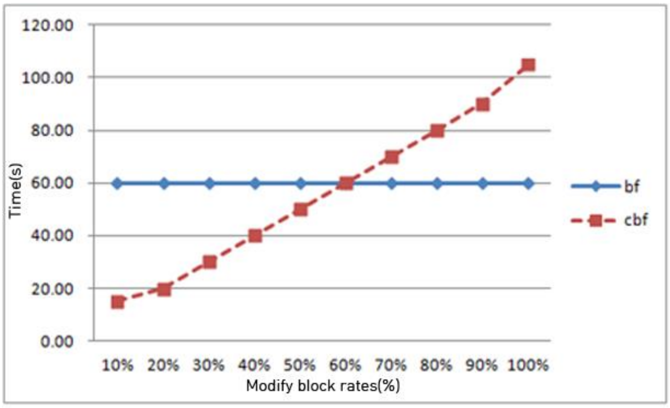

Figure 7. Update Time for Modifying Block Rates

Simulations were measured in real time as a file is added to the cloud. Each time 1.4GB was required, increasing the number of blocks to be corrected in the file. The size of each block was 1 MB. The false-positive rate of the filters used in the experiments had a value of 0.0001.

Figure 7 shows the results of the simulation. In the simulation, the filters have different update times for the modified data blocks. When using a Bloom Filter in PDP, it takes the same time each time, because of the need to create a filter for each blocks However, when using a Counting Bloom Filter, there is a time difference according to the ratio of the block.

Figure 8 shows the setting data of the blocks included in the query from the server-side to the measure the time it takes to generate a response statement. If the number of blocks increases, it is necessary to perform another operation to set the filters and the time is increased linearly. Since the insertion of the two run-time filters shows a slight difference in the performance of the response statement generation time between the two filters, two filter's result is similar.

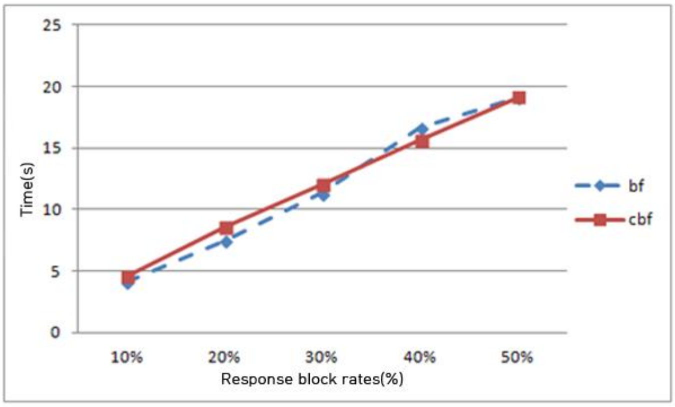

Figure 8. Time to Create Response Statements for Modified Block Rates

5. Conclusion

In this paper, we propose an extended PDP scheme for dynamic data in a cloud environment. Provable data possession based on metadata is created by the pre-process. The metadata must be maintained with respect to the most recently stored data. When the user modifies stored data in a cloud environment, PDP must update the data and generate new metadata. If each update-process performs repetitive operations, the calculation overhead for modifying the data increases. If the data frequently changes in this case, there is a problem of performance degradation. We performed simulations for our proposed scheme. The simulation result show that the proposed scheme updates faster than does the Bloom filter. When users update many blocks using proposed PDP, proposed PDP have to re-calculate for majority blocks. Therefore, this modification will be more effective than the partial result of the proposed method. Future studies will have to research techniques that can improve performance when the majority of the blocks have to be modified.

Acknowledgment

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(NRF-2012R1A1A2005293).

References

[1] S. J. Kim, "Information Security Plan on Cloud Computing - Information Security Management System", Korean review of management consulting, vol. 1, no. 8, **(2010)**, pp. 194-208.

[2] M. Y. Louk, H. T. Lim and H. J. Lee, "Security System for Healthcare Data in Cloud Computing", International Journal of Security and its Applications, vol. 8, no. 3, **(2014)**, pp. 241-248.

[3] K. C. Lee, "Security Threats in Cloud Computing Environments", International Journal of Security and its Applications, vol. 6, no. 4, **(2012)**, pp. 25-32.

[4] A. Shahzad and M. Hussain, "Security Issues and Challenges of Mobile Cloud Computing", International Journal of Grid and Distributed Computing, vol. 6, no. 6, **(2013)**, pp. 37-50.

[5] Y. G. Min, "Impediments and response measures for the cloud services activation", TTA Journal, vol. 125, **(2009)**, pp. 37-41.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable Data Possession at Untrusted stores", Proceedings of the 14th ACM conference on Computer and communications security, **(2007)**, pp. 598-609.

[7] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files", the 14th ACM conference On Computer and communications security, ACM, **(2007)**, pp. 584-597.

[8] Y. Zhu, H. Hu, G. Ahn and M. Yu, "Cooperative Provable Data Possession for integrity verification in multicloud storage", Parallel and Distributed Systems, IEEE Transactions on, vol. 23, no. 12, **(2012)**, pp. 2231-2244.

[9] X. Sun, L. Chen, Z. Xia and Y. Zhu, "Cooperative Provable Data Possession with Stateless Verification in Multicloud Storage", Journal of Computational Information Systems, vol. 10, no. 8, **(2014)**, pp. 3403-3411.

[10] B. R. Purshothama and B. B. Amberker, "Provable data possession scheme with constant proof size for outsourced data in public cloud", Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference on, **(2013)**, pp. 1662-1667.

[11] T. Shuang, C. Zhi-kun and Z. Jian-feng, "Data Blocks' Signature in Cloud Computing", International Symposium on Computational and Business Intelligence, **(2013)**, pp. 49-55.

[12] N. S. Cho and G. Y. Jang, "Technology Trends and Prospects in Homomorphic Encryption", Weekly Technology Trends, vol. 1522, **(2011)**, pp. 15-25.

[13] C. Li, Y. Chen, P. Tan and G. Yang, "An Efficient Provable Data Possession scheme with data dynamics", Computer Science & Service System (CSSS), **(2012)**, pp.706-710.

[14] B. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors", Communications of the ACM, vol. 13, no. 7, **(1970)**, pp. 422-426.

[15] T. Aditya, P. K. Baruah and R. Mukkamla, "Space-efficient Bloomfilters for Enforcing Integrity of Outsourced Data in Cloud Environments', the IEEE 4th International Conference on Cloud Computing, IEEE, **(2011)**, pp. 292-299.

[16] L. Fan, P. Cao, J. Almeida and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol", IEEE/ACM Transactions on Networking (TON), vol. 8, no. 3, **(2000)**, pp. 281-293.

[17] E. M. Jung, J. H. Jeong and Y. S. Hong, "A Novel Provable Data Possession Scheme using the Bloomfilter in Cloud Environment", Korea Computer Congress 2014, **(2014)**, pp. 955-957.

[18] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The Hadoop Distributed File System", Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium, **(2010)**, pp.1-10.

[19] S. W. Choi, S. H. Park, H. S. Choi and S. R. Yoon, "Optimization of Hadoop Environments for Small Files", IEIE Conference, the institute of electronics engineers of Korea, vol. 36, no. 1, **(2013)**, pp.1297-1300.

Authors

Eunmi Jung, She received the B.S. degree and M.S. degree from the Dept. of Computer Science Engineering, Dongguk University, Seoul, Korea, in 2013 and 2015. Her research areas include cloud storage security and distributed processing.

Junho Jeong, He received the B.S. degree from the Dept. of Computer Science, Dongguk University, Seoul, Korea, in 2007, and M.S. and Ph.D. degrees from the Dept. of Computer Engineering, Dongguk University, Seoul, Korea in 2009 and 2015, respectively. Currently, he is a Researcher of the Dept. of Computer Science and Engineering, Dongguk University, Seoul, Korea. His research areas include cloud security system, distributed processing, privacy security.