Journal of
Palestine Technical College
for Research & Studies

A Refereed Journal

Published by

Department of scientific Research
Palestine Technical College Deir El-Balah
Gaza Strip- Palestine

A Model to Detect the Integrity Violation of Shared File in the Cloud

Safaa T.Lulu

Tawfiq S.Barhoom

Information Technology College,
Islamic University, Gaza, Palestine

Abstract:

Many of cloud service providers pretend that user's data is saved securely with no data integrity violation. But as we know, anything saved in the internet environment can't be perfectly safe which reduce users trust that their files in safe situation, then they can't store sensitive data in the cloud.

In this work, we work on the problem of integrity of shared data over the cloud and a new model was built with dynamic data operation support. This model contains an intermediate system to detect file integrity violation by hashing the file, using SHA-256 algorithm, before saving it on the cloud and after retrieving it, then compare the two hash values.

Several experiments were conducted to ensure the ability of detecting data integrity violation, different sizes of .doc files starting from 13KB, ending with 5MB were collected. On the other hand, different types of tests were conducted to evaluate the accuracy and concurrent access. The evaluation process proves that our intermediate system is able to guarantee Files accuracy and detect many types of file modifications. Also, it is able to organize concurrent access operations on shared file and enable just one user to modify File at the same point.

Keywords: cloud computing, integrity detection, shared file, dynamic data operations, accuracy, concurrent access.

1. Introduction

Nowadays, there is an increasing trend in outsourcing data to remote cloud servers, where users outsource their files to remote cloud servers which may be controlled by untrustworthy cloud storage providers as anything saved on the internet can't be guaranteed at all. And there are some security risks involved with releasing control over user's data. One of them is the cloud service provider could intentionally or accidentally modify some data from the cloud server. Because the user does not know, how the cloud provider control or monitor his data (Ashdown, 2016).

So, it's vital to ensure that files are kept properly in a consistent way and file integrity is tracked. Surely, this advantage will be more useful when files are shared between a group of users, especially, because Integrity Violation Detection will be performed without the need to save a copy of the data locally. And will help in decreasing organizations fear toward outsourcing their files to cloud. On the other hand, it could be a good way of File Archiving, as we can retrieve them after a long period of time and detect their integrity violation easily.

The Files are the most common type used in data sharing and used by different types of users regardless to their technology knowledge. So, the proposed

model is designed to consider the shared file.

1.1. Data Integrity:

It is defined as the consistency and accuracy of stored data in absence of any alteration to the data between two updates of a file or record. (businessdictionary,2016). In this work, the model is to detect the integrity violation of shared file on the cloud, as the integrity of the file content is a very important to have a right information and as consequence has a right decision.

1.2. Hashing process:

Hashing is the process of transformation of a string of characters into a fixed-length value that represents the original string (Rouse, M. ,2005). There are many different hashing algorithms such as MD2, MD5, SHA...etc.

We go to the most common hash algorithm family SHA (standing for secure hash algorithm), there are several versions of SHA: SHA0, SHA1, SHA2 and finally SHA3 introduced in 2012 (Ibnu, 2016).

In this work, SHA256 algorithm will be used in this model, as it is a strong hash algorithm, resistant to collision attacks and isn't broken by any attack.

2. Related Works

In cloud environment, large numbers of threats are raised. Data integrity is one of the main threats. A lot of researchers focused on providing data correctness within the cloud and introduce many answers to decrease the threat related to data integrity among the cloud.

Kavuri et al. (Kavuri et al., 2014) proposed an improved hash based message integrity verification process to ensure the confidentiality of the user's information in the remote storage, users can access the required files using their identity along with the message integrity value. Most of the existing hash methods integrate hash value to the original data. But Kavuri et al. (Kavuri et al.,2014) method is to store both the encrypted data and its hash value separately in the cloud storage.

Kumar and Saxena (Kumar, S.; Saxena, A., 2011) proposed a scheme which gives a proof of data integrity in the cloud by encrypting only few bits of data per data block not the whole data, so reduce computational overhead on the clients also the client storage overhead is minimized as it does not store any data with it. But this scheme cannot be applied when the data need to be dynamically changed.

These works tried to detect integrity in the cloud but in different ways. But they didn't support Dynamic Operations or File Sharing.

Some Researchers enable a third-party verifier to check data integrity, like Lianhong and Hua (Lianhong, Z.; Chen,H., 2010) and Subramanian

(Subramanian, R.; Kumar, S. ,2011). Regardless supporting Dynamic Operations or File Sharing, these works supported third party verifier, which could be a real security risk for data Integrity in the cloud.

Few of researchers deal with integrity problem taking in consideration data sharing, Marimuthu et al. (Marimuthu,K. et al., 2014) proposed a secure multi-owner data sharing scheme, for dynamic group in the cloud and they integrated Image based authentication with it to achieve high level of security in authenticating the use.

Also, Aarthi and Indira (Aarthi, 2016) proposed a methodology that provides data sharing. If user wants to share data, encrypted data is read from cloud database, then perform under re-encryption by shared user encryption key, because a unique encryption key is maintained for each user. Then that re-encrypted data stored in cloud database server. But when a user want to access data, he/she should sends a download request to the third part (that is responsible for key management, encryption, decryption, and access control) which could be one of threats on data. So, Marimuthu et al. (Marimuthu,K. et al., 2014) and Aarthi and Indira (Aarthi, 2016) used different types of hashing and encryption and supported File Sharing but didn't support dynamic operations on these files.

Few of works consider dynamic data files and support dynamic data operation on it, Barhoom and Saqallah, (Barhoom, D. T. a. S., Zakaria Kh, 2014) proposed a model with dynamic data operations support including data append, update and delete, to ensure the integrity of user's data in the cloud based on verification tokens and erasure- codeddata without the need to keep the data locally at the client side. This model reduces the computational overhead of the client but it doesn't ensure the integrity of a shared data by a group of users.

Khatri and Jethava (Khatri, T. ; Jethava, G., 2013) proposed a scheme that provides data integrity checking frequently and supports dynamic operation using Merkle Hash Tree construction. But also, it doesn't detect integrity of shared data and another shortcoming is that used SHA-1 hash algorithm which is broken.

Although Khatri and Jethava (Khatri, T. ; Jethava, G., 2013) and Barhoom and Saqallah (Barhoom, D. T. a. S., Zakaria Kh, 2014) supported Dynamic Operations on files stored in the clouds of their models, but they didn't apply integrity detection on shared files.

Finally, after an advance step of working, we found GitHub, which is a web-based version-control and collaboration platform for software developers (Rouse, M. ,2016).

GitHub can actually be used for any types of files (including word documents) so if you have a team that is constantly making changes to a document, you can actually use GitHub as your version control system which support sharing data with dynamic operations on it. Also it keeps track of all the changes that have been pushed to the repository (as a log file). While GitHub is a software service, our work is specialized for files. but it is a public cloud and we should take in consideration, trust issues especially when file contents are sensitive and can't assume any error.

3. The Proposed Model

In this work, Integrity detection violation includes two areas: fears from outside the cloud (as internet is not totally safe) and fears from not-authorized use from internal users.

Our model contains an intermediate system to detect file integrity violation by hashing the file before saving it on the cloud and after retrieving it, then compares the two hash values. The proposed model supports dynamic data operation on shared files, including append , update and delete. Also, we focus on how to keep dynamic operations working in concurrency way, so no user can coflict the work of another user on a shared file.

Figure 1. The Proposed Model Architecture

Figure(1) presents our proposed model which includes the following:

Users: One admin and other users, each one has his own account in the system.

Login module: Each user log in with his own account, (all users in the group except admin are with same privileges).

Hashing Module: Before storing the file in the cloud (F), a hash method (SHA256) will be used to hash the file content. Then, hash value will be stored on the intermediate system database with the file ID, time of storing and file creator.

Permission module: In principle, each user will have the same permissions on file, but when a service (File) is downloaded by a user (to update it), no other user can update it till this user or the file creator re-activate the update property or upload an updated version of the file. If none of them (user or file creator) activated the update property, the system will activate it automatically after a pre-specified period of time, in our prototype, the pre-specified period is 15 minutes. In future prototype, Admin can specify the period as he wants and when he wants.

Log file module: To map the change on the file, intermediate system will contain a log file, so it saves every dynamic operation type (insert, delete or update), its actor identity and time of operation so if any file modification operation will be stored, so every user will be responsible of the operation he/she made on this file.

Integrity Check: To check the correctness of the file data in the cloud, we will retrieve a stored file, from the cloud, then, the hash value of the retrieved file will be re-calculated and finally, compare it with shared hash value that stored in the intermediate system. If the two hash values are matched, then, there is no problem. If not, so Notification alert system will sends an alert to the user to know that the file integrity of the file was violated.

Notification Alert: sends alert to the user if data integrity of the file is violated.

Dynamic Data Operation Support: Users can perform dynamic data operations (append, update or delete to modify the data file) under the restriction of this file permissions. For example, if a user open a File for writing (updating), no one can write on it at that time. To delete a file, any user can delete it, actually it is not deleted, it is hidden to users, and only admin can see it and confirm deletion or restore it to the system.

4. Implementation

In this work, a model to detect the integrity violation of shared file in the cloud is introduced. Also it includes dynamic data operations support on shared files (including data update, append and delete).

An intermediate system is built in PHP 5 Language to detect file integrity violation by hashing the file before saving it on the cloud and after retrieving it, then compare the two hash values.

Also, we will focus on how to keep dynamic operations working in concurrency way, so no user can conflict the work of another user on a shared file. The presented flow chart in Figure (2), illustrates how the intermediate system works to detect integrity violation of shared files on the cloud.

Figure 2. Intermediate System Flow Chart

When a user wants to store (upload) a file in the cloud (F), a hash method (SHA256) is applied to hash the file content. Then, hash value will be stored on the intermediate system database with the file ID, time of storing and file creator.

In principle, each user have the same permissions on file, but when a service (File) is downloaded by a user (to update it), no other user can update it till this user or the file creator activate the update property or upload an updated version of the file. If none of user or file creator activate the update property, the system will activate it automatically after a pre-specified period of time, in our prototype, the pre-specified period is 15 minutes.

To map the change on the file, intermediate system contains a log file, so it

saves every dynamic operation type (insert, delete or update), its actor identity and time of operation.

To check the correctness of the file data in the cloud, when a user retrieves a stored file from the cloud, then, the intermediate system calculates the hash value of the retrieved file and finally, compare it with old hash value that stored in the intermediate system database. If the two hash values are matched, then, there is no problem. If not, so Notification alert system will send an alert to the user to know that the file integrity is violated.

5. Experiments

Several experiments were conducted on our implementation to demonstrate its ability to detect data integrity violation, and to evaluate its accuracy, by enabling a group of users to share one file many times consecutively and detect the integrity violation of this file at each time. Also, concurrent access is evaluated by enabling multiple users to access one file at the same time.

Experimental Environment and Tools:

Our experiments are conducted with a laptop (msi), running windows 8 x86 operating system with core i5, with 8 GB memory RAM. PHP 5 (Laravel Framework) is used as a programming language

5.1. Setup a Private Cloud:

We built an intermediate system that enables users to upload their files, update, read or delete these files. This intermediate system is hosted on an Intel(R) Xeon(R) server with CPU E3-1231 v3 32 Gb ram.

5.2. Dataset:

To accomplish the experiments, a dataset must be collected, so, different sizes starting with 13 KB, ending with 5MB were chosen. SHA-256 hash algorithm was applied on these files.

As a result: it was concluded that for any different size and there is no difference between hashing time of files despite their different sizes.

5.3. Results of experiments:

Four different types of experiments have been constructed:

Experiment 1: Detect files integrity violation after intended hacking: try to modify 20 files content without modifying hash value.

Result: The intermediate system didn't allow users to update these 20 hacked files and it displayed a warn message to inform the user that these files integrity is VIOLATED!!!

Experiment 2: Update same file many times consecutively: to test if hash value is re-calculated after each update with no conflicts.

Result: This experiment approved that a file can be shared and updated

consecutively by number of users with no conflicts or problems.

Experiment 3: Update same file concurrently: to test if model will allow more than one user to access the same file. This experiment was divided into two experiments: (file creator is one of the three users, file creator isn't one of the three users), because this model has a property that gives a priority to the file creator to update first.

Result: Intermediate system enables one and only one user to update the file at the same time.

Experiment 4: Hacking by different types of modification: modify very small parts of file contents to test if hash valued changed or not.

Result: After applying this experiment and trying to access these files, a warn message was appeared that File is violated!

6. Evaluation of the Model

The most important measure in our model is accuracy, it refers to whether the data correctly records the business object or event it represents. It has two requirements: it must be the right value and it must represent the value in a consistent form with all other representations of the same value (Olson, 2002).

It is measured by enabling a group of users to share one file many times consecutively and detect the integrity violation of this file at each access. The other measure is to evaluate the concurrent access, by enabling multiple users to access the shared file at the same time to detect any file access conflict.

6.1. Accuracy Evaluation:

Three experiments were conducted, in order to evaluate the accuracy:

1. Update one file directly in the server side for many times: (**normal way**)
 - a. Change some bytes in the file (one word, one line, one paragraph).
 - b. Insert some bytes to the file.
 - c. Delete some bytes from the file.
2. Also, we repeat the previous experiment but in an illegal way (**Hacking**), mean to update any file without re- computing a new hash value.

Collision resistant: We checked our dataset SHA256 hash value for all stored files (more than 100 file), after files were hashed. We found that SHA256 is resistant to collision attacks, as there isn't any matching between two files hash values.

After conducting all these tests, we found that this intermediate system is able to guarantee Files accuracy and detected all type of file modifications.

6.2. Evaluate concurrent access:

concurrent access refers to the assurance that users can access data at the same time (Ashdown and Kyte, 2016). Our model enables just one user to update

File at the same time, so when multiple users try to update one file at the same time, the intermediate system should enable just one user and stop others.

To ensure this concurrent access, the following experiment was conducted many times:

1. More than one user (about three users, sometimes four users) tried to access a shared file in the cloud to update it.

7. Intermediate system enables just one user to access it, and notify others that a user is updating this File now by disabling update property temporary. And it is able to organize concurrent access operations on shared file and enables just one user to modify File at the same point.

7. Conclusion

2. A model to detect the integrity violation of shared files in the cloud with dynamic operation support has been built. This model contains an intermediate system to detect file integrity violation by hashing parts of the file, using SHA-256 algorithm, before saving it on the cloud and after retrieving it, then

compare the two hash values, if they are matched, then there is no problem, else integrity is violated and alert message is appeared.

Several experiments were conducted with different (.doc) files sizes (from 13KB to 5MB) to ensure the ability of detecting data integrity violation, and to evaluate the accuracy and concurrent access.

The evaluation process proves that this model has been performed at an acceptable level and is highly efficient against malicious data modification attack.

8. Future work

As future challenges the research approach can be updated through adding the following:

1. Support more different types of files (image, voice, and video).
2. Make a periodically check on less used files in the cloud, to detect their integrity violation.
3. Work on other types of data such as structured and tabular data.
4. Enlarge work scope to include error localization and error recovery.

References

1. Aarthi, N. Indira. (25 Feb 2016) *Enabling Efficient and Protected Sharing of Data in Cloud Computing*. Paper presented at the International Conference On Information Communication And Embedded System (ICICES 2016).
2. Ashdown, L.(2016).*Concurrency*. Retrieved Dec. 30,2016 from <https://docs.oracle.com/database/121/CNCPT/consist.htm#GUID-7AD41DFA-04E5-4738-B744-C4407170411C>

3. Barhoom, D. T. a. S., Zakaria Kh. (2014). "*A Model to Ensure the Integrity of User's Data in the Cloud.*". Islamic Univercity. businessdictionary. (2016). Data integrity. 2016, from <http://www.businessdictionary.com/definition/data-integrity.html> Ibnu (2016).SHA. Retrieved Dec. 10, 2016 from <https://steemit.com/steemit/@ibnu/nice-selection-of-steemit-in-choosing-256-as-a-hash-algorithm>
4. Kavuri, S.; Kancherla, S.; Rao, G. and Rao,B (27 Sep 2014). "*Data Authentication and Integrity Verification Techniques for trusted/Untrusted Cloud Servers.*" Paper presented at the International Conference on Advances in Computing, Communications and Informatics (ICACCI).
5. Khatri, T. ; Jethava, G. (2013). *Improving Dynamic Data Integrity Verification in Cloud Computing.*
6. Kumar, S.; Saxena, A. (4 Jan 2011). "*Integrity Proofs in Cloud Storage.*". Paper presented at the Communication Systems and Networks, Third International Conference.
7. Lianhong, Z.; Chen,H. (2010). "*Secuirty Storage in the Cloud Computing: A RSA-based Assumption Data.*". Paper presented at the International Coriference on Educational and Information Technology.
8. Marimuthu,K.; Sashi, K.; Setty,S.; Tainwala, K. (10 May 2014). "*Scalable and Secure Data Sharing for Dynamic Groups in Cloud.*" Paper presented at the IEEE International Conference on Advanced Communication Control and Computing Technologies (ICACCCT).
9. Al-Mourad, M. H. a. (2014). "*Effective Third Party Auditing in Cloud Computing.*". Advanced Information Networking and Applications Workshops (WAINA).
10. Olson, J. (2002). *Accuracy.* Retrieved Dec. - 23, 2016, from <http://www.information-management.com/infodirect/20021108/6019-1.html>
11. Rouse, M. (2005).*Hashing.* Retrieved 15-April, 2016, from <http://searchsqlserver.techtarget.com/definition/hashing>
12. Rouse, M. (2016). *Github.* Retrieved Dec. 14, 2016 from

<http://searchitoperations.techtarget.com/definition/GitHub>

13. Subramanian, R.; Kumar, S. (2011). An Efficient and Secure Protocol for Ensuring Data Storage Security in Cloud Computing ". *IJCSI International Journal of Computer Science Issues*. 35 (9), p8