ISSN 2319-2720

Volume 7, No.2, April – June 2018

International Journal of Computing, Communications and Networking

Available Online at http://www.warse.org/ijccn/static/pdf/file/ijccn56722018.pdf

https://doi.org/10.30534/ijccn/2018/56722018

MEMORY AND SPACE EFFICIENT DATA INTEGRITY CHECKING SCHEME ON REAL TIME CLOUDS

Dhanraj[1] Chandana S[2], Puneeth A[3], Punith Raj N[4] , Rohith M B[5]

[1] Assistant professor EWIT, India, dhanraj@ewit.edu

[2]EWIT, India, chandanagowda803@gmail.com

[3] EWIT, India, Puneeth.sp013@gmail.com

[4] EWIT, India, punithraj.n4122@gmail.com

[5] EWIT, India, rohitmb1995@gmail.com,

ABSTRACT

As an important application in cloud computing, cloud storage offers user scalable, flexible, and high-quality data storage and computation services. A growing number of data owners choose to outsource data files to the cloud. Because cloud storage servers are not fully trustworthy, data owners need dependable means to check the possession for their files outsourced to remote cloud servers. To address this crucial problem, some remote data possession checking (RDPC) protocols have been presented. But many existing schemes have vulnerabilities in efficiency or data dynamics. In this project, we provide a new efficient RDPC protocol based on homomorphism hash function. The new scheme is provably secure against forgery attack; replace attack, and replay attack based on a typical security model. To support data dynamics, an operation record table (ORT) is introduced to track operations on file blocks. We further give a new optimized implementation for the ORT, which makes the cost of accessing ORT nearly constant. Moreover, we make the comprehensive performance analysis, which shows that our scheme has advantages in computation and communication costs. Prototype implementation and experiments exhibit that the scheme is feasible for real applications

Key words: KDPC, ORT**,** Reply attack, Cloud

1. INTRODUCTION

Cloud computing emerges as a novel computing paradigm subsequent to grid computing. By managing a great number of distributed computing resources in Internet, it possesses huge virtualized computing ability and storage space [1]. Thus, cloud computing is widely accepted and used in many real applications [2]. As an important service for cloud computing, cloud service provider supplies reliable, Manuscript received April 2, 2016; revised June 24, 2016 and August 7, 2016;

accepted August 7, 2016. Date of publication August 17, 2016; date of current version October 31, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61272542, Grant 61672207 and Grant 61300213, in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions, Jiangsu Provincial Natural Science Foundation of China under Grant BK20161511 and in part by the Fundamental Research Funds for the Central Universities under Grant 2016B10114, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chip-Hong Chang. H. Yan, J. Li, and Y. Zhang are with the College of Computer and Information, Hohai University, Nanjing 211100, China (e-mail: pxy_hao@163.com;ljg1688@163.com; zyc_718@163.com). J. Han is with the Jiangsu Provincial Key Laboratory of E-Business, Nanjing University of Finance and Economics, Nanjing 210003, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: jghan22@gmail.com). It provides the users with a more flexible way called payas-you-go model to get computation and storage resources on-demand. Under this model, the users can rent necessary IT infrastructures according to their requirement rather than buy them.

Thus, the up-front investment of the users will be reduced greatly. In addition, it is convenient for them to adjust the capacity of the rented resource while the scale of their applications changes. Cloud service provider tries to provide a

promising service for data storage, which saves the users costs of investment and resource. Nonetheless, cloud storage also brings various security issues for the outsourced data.

Although some security problems have been solved [3–10], the important challenges of data tampering and data lost still exist in cloud storage. On the one hand, the accident disk error or hardware failure of the cloud storage server (CSS) may cause the unexpected corruption of outsourced files. On the other hand, the CSS is not fully trustworthy from the perspective of the data owner, it may actively delete or modify files for tremendous economic benefits. At the same time, CSS may hide the misbehaviors and data loss accidents from data owner to maintain a good reputation. Therefore, it is crucial for the data owner to utilize an efficient way to check the integrity for outsourced data. Remote data possession checking (RDPC) [11] is an effective technique to ensure the integrity for data files stored on CSS. RDPC supplies a method for data owner to efficiently verify whether cloud service provider faithfully stores the original files without retrieving it. In RDPC, the data owner is able to challenge the CSS on the integrity for the target file. The CSS can generate proofs to prove that it keeps the complete and uncorrupted data. The fundamental requirement is that the data owner can perform the verification of file integrity without accessing the complete original file. Moreover, the protocol must resist the malicious server which attempts to verify the data integrity without accessing the complete and uncorrupted data [12]. Another desired requirement is that dynamic data operations should be supported by the protocol. In general, the data owner may append, insert, delete or modify the file blocks as needed. Besides, the computing complexity and communication overhead of the protocol should be taken into account for real applications.

2. RELATED WORKS

The first RDPC was proposed by Deswarte et al. [11] based on RSA hash function. The drawback of this scheme is that it needs to access the entire file blocks for each challenge. In 2007, the provable data possession (PDP) model was presented by Ateniese et al. [13], which used the probabilistic proof technique for remote data integrity checking without accessing the whole file. In addition, they supplied two concrete schemes (S-PDP, E-PDP) based on RSA. Although these two protocols had good performance, it's a pity they didn't support dynamic operations. To overcome this shortcoming, in 2008, they presented a dynamic PDP scheme by using symmetric encryption [14]. Nonetheless, this scheme still did not support block insert operation. At the same time, lots of research works [15]–[19] devoted to construct fully dynamic PDP protocols. For instance, Sebé et al. [15] provided a RDPC protocol for critical information infrastructures based on the problem to factor large integers, which is easily adapted to support data dynamics.

3. METHODLOGY

PROBLEM STATEMENT

Cloud storage servers are not fully trustworthy; hence data owners need dependable means to check the possession for their files stored on the remote cloud servers. To address this problem, some remote data possession checking (RDPC) protocols have been presented. But many existing schemes have vulnerabilities in efficiency or data dynamics

Module 1: Implementation of user profile operations
Module 2: Implementation of Hashing technique for performing data possession
Module 3: Implementation of Data models operations
Module 4: Implementation of Data access operations
Module 5: Implementation of Proof for hashing properties
Module 6: Live cloud deployment

4. MOTIVATION

It is essential for data owners to verify the integrity for the data stored on CSS before using it. For example, a big international trading company stores all the imports and exports record files on CSS. According to these files, the company can get the key information such as the logistics quantity, the trade volume etc. If any record file is discarded or tampered, the company will suffer from a big loss which may cause bad influence on its business and development. To avoid this kind of circumstances, it is mandatory to check the integrity for outsourced data files. Furthermore, since these files may refer to business secret, any information exposure is unacceptable.

If the company competitor can execute the file integrity checking, by frequently checking the files they may obtain some useful information such as when the file changes, the growth rate of the file etc, by which they can guess the development of the company.

Thus, to avoid this situation, we consider the private verification type in our scheme, that is, the data owner is the unique verifier. In fact, the current research direction of RDPC focuses on the public verification, in which anyone can perform the task of file integrity checking with the system public key. Although RDPC with public verification seems better than that with private verification, but it is unsuitable to the scenario mentioned above.

5. PRELIMINARIES

In this section, we introduce the preliminary knowledge used throughout this paper. A. Homomorphic Hash Function Inspired by [19] and [21], our scheme adopts the homomorphic hash function defined in [20] as the basis, which is described as

First, the algorithm HKeyGen(λp,λq,m,s) → K is utilized to obtain the homomorphic key. It takes four security parameters as inputs, in which λp and λq are two discrete log security parameters, m is the sector count of the message and s is a random seed. It outputs the homomorphic key K = (p,q, g), where and q are two random big primes with the properties of |p|=λp, |q|=λq and q|(p −1), g =[g1,g2,⋯,gm] is a 1 × m row-vector composing of random values in Z∗ p with order q. The detailed process of this algorithm is shown in Fig.1, in which the function f(x) is the pseudo-random number generator with seed s and outputs the next number in its pseudo-random sequence, scaled to the range {0,⋯,x −1}

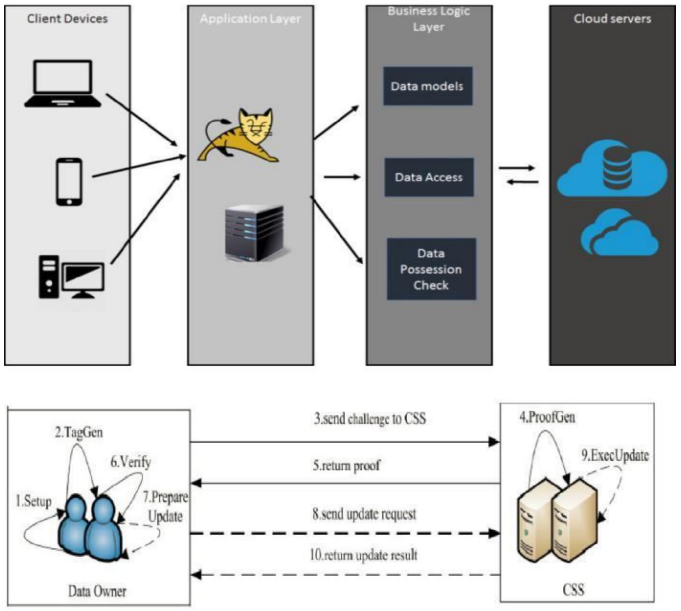
[19]. The computation of the hash value X of the message S represented by H(S) → X is defined as following. S is divided into m sectors S = (s1,s2,⋯,sm), the hash value is calculated as: HK(S) = m i=1 gsi i mod p. Homomorphic key generation algorithms. messages Si and Sj, whereSi = (s1i,s2i,⋯,smi), Sj = (s1j,s2j,⋯,smj), we defineSi + Sj = (s1i + s1j,s2i + s2j,⋯,smi + smj) mod q. The homomorphic property of H(·) can be verified by: H(Si +Sj) = m t=1 gsti+stj t mod p = m t=1 gsti t mod p· m t=1 gstj t mod p= H(Si)·H(Sj).

B. Operation Record Table Refer to [18] and [25], to support dynamic operations on file blocks, we introduce a simple flexible data structure named operation record table (ORT). The table is reserved on the data owner side and used to record all the dynamic behaviors on file blocks. ORT has a simple structure with only three columns, that is Block Position (BP), Block Index (BI) and Block Version (BV). The BP represents the physical index for the current block in the file, normally its value is incremented by 1. The BI represents the logical index for the current block, which is no necessary equal to BP but relevant with the time when the block appears in the file. The BV indicates the current version for the block. If the data file is initially created, the BV values for all blocks are 1. When one concrete block is updated, its BV value is incremented by 1. It is noted that using the ORT table will increase the storage overhead of the data owner by O(n), where is the count of blocks. However, this extra storage cost is very little. For example, a 1GB-file with 16KB block size only needs 512KB space to store an ORT realized by linked list (<0.05% of the file size).

C. Outline of Our RDPC Protocol In this paper, we investigate the cloud storage system including two participants: CSS and data owner. The CSS has powerful storage ability and computation resources, it accepts the data owner's requests to store the outsourced data files and supplies access service. The data owner enjoys CSS's service and puts large amount of files to CSS without backup copies in local. As the CSS is not assumed to be trustable and occasionally misbehave, for example, modifying or deleting partial data files, the data owner can check the integrity for the outsourced data efficiently. A RDPC scheme includes the following seven algorithms: KeyGen(1k,λp,λq,m,s)
→(K,sk). The data owner executes this algorithm to initialize the system and generate keys.

6. SYSTEM ARCHITECTURE DIAGRAM

The below figure shows a general block diagram describing the activities performed by this project. The entire architecture has been implemented in nine modules which we will see in high level design and low level design in later chapters.



Major divisions in this project are

Data Access Layer

Data access layer is the one which exposes all the possible operations on the data base to the outside world. It will contain the DAO classes, DAO interfaces, POJOs, and Utils as the internal components. All the other modules of this project will be communicating with the DAO layer for their data access needs

Account Operations

Account operations module provides the following functionalities to the end users of our project.

- Register a new seller/ buyer account
- Login to an existing account
- Logout from the session

- Edit the existing Profile
- Change Password for security issues
- Forgot Password and receive the current password over an email
- Delete an existing Account

Account operations module will be re-using the DAO layer to provide the above functionalities.

7. HOMOMORPHIC HASH FUNCTION

Homomorphic Hashing provides a neat solution to the problem of verifying data from untrusted peers when using a fountain coding system, but it's not without its own drawbacks. It's complicated to implement and computationally expensive to compute, and requires careful tuning of the parameters to minimize the volume of the hash data without compromising security. Used correctly in conjunction with fountain codes, however, Homomorphic Hashing could be used to create an impressively fast and efficient content distribution network.

Write Data Models

Here the end users will be provided with an interface where he/she can write the data models for the data he/she is planning to write on the cloud storage. Data models defines the structure of the data. Typically the data models comes in the form of key-value pairs. The idea of introducing the concept of data model is to generalize the structure of the data to be stored on the cloud. Different organizations can create different data models based on the nature of the data they have to be written on to the clouds. This enables the organizations from multiple domains to use our project independent of the type of the data they have.

Read Data model

Here, the end users will be able to see the list of all the data models created by him/her in the previous section. The end user will also be able to perform the delete operation on the data models he/she have created. The data models written in the previous section will be persisted in the MySQL database and can be accessed back in this module using the data model DAO layer.

Write Data

This module enables the end user to perform the data write operation to the cloud storage space. The data will be persisted in the MySQL instance of the cloud application deployed in any of the cloud service provider. The customer first will have to select the data model indicating the type of the data to be written on to the cloud. After selecting the data model, the user will be provided with an interface where they provide the key value pairs for the data models. The data written by the user will first be encrypted using the AES (Advanced Encryption Standard) cryptography and then the hash code of the encrypted data will be found out using the homomorphic hash function. The hash code of the encrypted data will be persisted locally, and the actual encrypted data will be persisted in the MySQL instance on the cloud storage space.

Access Data

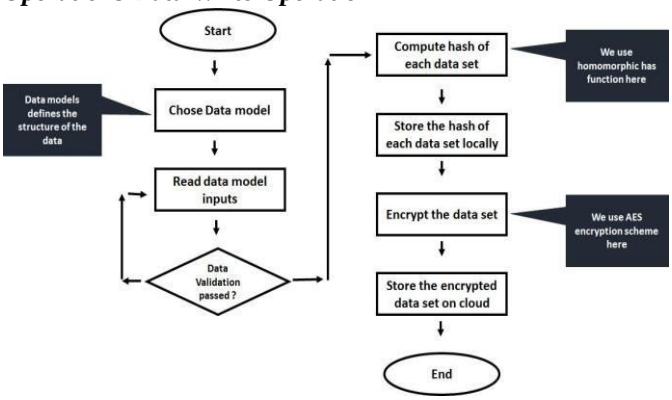
Here the end users will be able to see the list of all the data they had written on to the cloud storage space in the previous section. The list of all the data will be retrieved from the MySQL instance of the cloud storage space and then be displayed on the HTML interface. The user can then perform the data decryption of the data using the Advanced Encryption Standard (AES) cryptography. The user will then be able to perform the data update operation which will again make a call to the homomorphic hash function. The user can then be able to perform the data possession check.

The hash code of the encrypted data stored locally will be compared with the hash code of the encrypted data from the cloud storage. If those two of the hash codes differs, then the possession check will be failing indicating that the data has been illegally modified by the cloud service provider, else, the data possession check will be passing indicating that the data is left untouched on the cloud storage space.

The DAO layer is the service layer which provides database CRUD (create, update, read, and delete) services to the other layers.It will contain the POJO classes to map the database tables into java object. It will also contain the Util classes to manage the database connections. Homomorphic Hashing provides a neat solution to the problem of verifying data from untrusted peers when using a fountain coding system, but it's not without its own drawbacks. It's complicated to implement and computationally expensive to compute, and requires careful tuning of the parameters to minimize the volume of the hash data without compromising security. Used correctly in conjunction with fountain codes, however, Homomorphic Hashing could be used to create an impressively fast and efficient content distribution network. Here the end users will be provided with an interface where he/she can write the data models for the data he/she is planning to write on the cloud storage. Data models define the structure of the data. Typically the data models come in the form of key-value pairs. The idea of introducing the concept of data model is to generalize the structure of the data to be stored on the cloud. Different organizations can create different data models based on the nature of the data they have to be written on to the clouds. This enables the organizations from multiple domains to use our project independent of the type of the data they have.

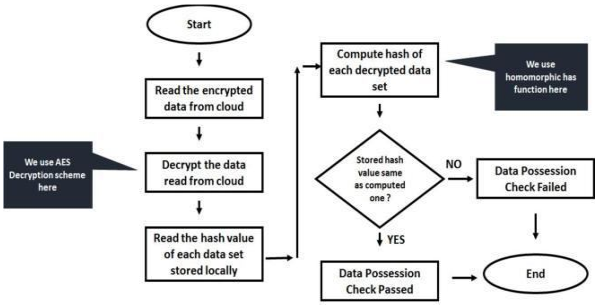
Also, the end users will be able to see the list of all the data models created by him/her in the previous section. The end user will also be able to perform the delete operation on the data models he/she have created. The data models written in the previous section will be persisted in the MySQL database and can be accessed back in this module using the data model DAO layer.

Module 4: Implementation of Data Access Operations Data Write Operation



This module enables the end user to perform the data write operation to the cloud storage space. The customer first will have to select the data model indicating the type of the data to be written on to the cloud. After selecting the data model, the user will be provided with an interface where they provide the key value pairs for the data models. The data written by the user will first be encrypted using the AES (Advanced Encryption Standard) cryptography and then the hash code of the encrypted data will be found out using the homomorphic hash function. The hash code of the encrypted data will be persisted locally, and the actual encrypted data will be persisted in the MySQL instance on the cloud storage space.

7.1 DATA READ OPERATION



Here the end users will be able to see the list of all the data they had written on to the cloud storage space in the previous section. The list of all the data will be retrieved from the MySQL instance of the cloud storage space and then be displayed on the HTML interface. The user can then perform the data decryption of the data using the Advanced Encryption Standard (AES) cryptography. The user will then be able to perform the data update operation which will again make a call to the homomorphic hash function. The user can then be able to perform the data possession check. The hash code of the encrypted data stored locally will be compared with the hash code of the encrypted data from the cloud storage. If those two of the hash codes differs, then the possession check will be failing indicating that the data has been illegally modified by the cloud service provider, else, the data possession check will be passing indicating that the data is left untouched on the cloud storage space.

8. CONCLUSION

As the saying goes “Necessity is the mother of all inventions”, a need for software which would control process and devices was recognized. Accordingly a highly interactive user friendly module based embedded technology with microcontrollers was developed to solve the problem. The module which is developed will make the job of process easier. The user module has resulted in reducing work of human also makes more comfortable.The module is, therefore functioning as a very good tool. Incorporating the future enhancement as specified earlier would make the software a perfect tool, which would help the user.

9. ACKNOWLEDGEMENT

We wish to offer our sincere gratitude to our principal Dr. K Channakeshavalu, Principal, EWIT, Bangalore, for his moral support towards completing my project work. We would like to thank Dr. Arun Biradar, Head of Department, Computer Science & Engineering, EWIT, Bangalore, for his valuable suggestions and expert advice. We deeply express my sincere gratitude to my guide Prof. Dhanraj , Assistant professor Department of CSE, EWIT, Bangalore, for his able guidance throughout the project work and guiding me to organize the report in a systematic manner. We thank our Parents, and all the Faculty members of Department of Computer Science & Engineering for their constant support and encouragement.

10. REFERENCES

[1] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, “Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,” Future Generat. Comput. Syst., vol. 25, no. 6, pp. 599–616, 2009.

[2] H. Qian, J. Li, Y. Zhang, and J. Han, "Privacy-preserving personal health record using multi-authority attribute-based encryption with revocation," Int. J. Inf. Secur., vol. 14, no. 6, pp. 487–497, 2015.
https://doi.org/10.1007/s10207-014-0270-9

[3] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," IEEE Trans. Services Comput., to be published, doi: 10.1109/TSC.2016.2520932.
https://doi.org/10.1109/TSC.2016.2520932

[4] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," IEEE Trans. Services Comput., to be published, doi: 10.1109/TSC.2016.2542813.
https://doi.org/10.1109/TSC.2016.2542813

[5] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," Int. J. Commun. Syst., to be published, doi: 10.1002/dac.2942.
https://doi.org/10.1002/dac.2942

[6] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," IEEE Trans. Parallel Distrib. Syst., vol. 23, no. 11, pp. 2150–2162, Nov. 2012.
https://doi.org/10.1109/TPDS.2012.50

[7] Z. Fu, X. Sun, Q. Liu, L. Zhou, and J. Shu, "Achieving efficient cloud search services: Multi-keyword ranked search over encrypted cloud data supporting parallel computing," IEICE Trans. Commun., vol. E98-B, no. 1, pp. 190–200, 2015.
https://doi.org/10.1587/transcom.E98.B.190

[8] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 9, pp. 2546–2559, Sep. 2016, doi: 10.1109/TPDS.2015.2506573.
https://doi.org/10.1109/TPDS.2015.2506573

[9] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multikeyword ranked search scheme over encrypted cloud data," IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 2, pp. 340–352, Feb. 2016.
https://doi.org/10.1109/TPDS.2015.2401003

[10] Y.-J. Ren, J. Shen, J. Wang, J. Han, and S.-Y. Lee, "Mutual verifiable provable data auditing in public cloud storage," J. Internet Technol., vol. 16, no. 2, pp. 317–323, 2015.

[11] Y. Deswarte, J.-J. Quisquater, and A. Saïdane, "Remote integrity checking," in Proc. 6th Work. Conf. Integr. Int. Control Inf. Syst. (IICIS), 2003, pp. 1–11.

[12] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," IEEE Trans. Knowl. Data Eng., vol. 23, no. 9, pp. 1432–1437, Sep. 2011.
https://doi.org/10.1109/TKDE.2011.62

[13] G. Ateniese et al., "Provable data possession at untrusted stores," in Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS), 2007, pp. 598–609.
https://doi.org/10.1145/1315245.1315318