Achieving Authentication of Remotely Stored Data

Vidyullata Devmane, B.K.Lande, Jyoti Joglekar, Hemant Kothari

Abstract: ***Data storage over cloud is a demanding service but it is vulnerable for various attacks. It is required to provide strong security to this data. An authenticity of the data on remote storage is necessary but data owners have to trust on cloud storage for their outsourced data. There is a need of some mechanism to check correctness of the data without compromising its confidentiality. We have studied homomorphic algorithms so that can be applied on the cipher text. Aim of the framework designed here is to enable any authentic user or auditor to verify hash value generated by owner with the hash value of encrypted data which is stored at remote storage for assuring authentication of the data. This can be done with strong mathematical cryptographic algorithms those can be used in the public environment and can provide better performance complexity.***

Keywords: Cloud storage data authenticity, Homomorphic algorithms, Hash tree.

I. INTRODUCTION

In the cloud environment, there is need of verification of intentional or unintentional modifications of data as cloud storages are managed by the service providers. Cloud storages give low cost ownership along with freedom to clients to use it as per requirement. But client will not be sure about the data intactness of outsources data; there are chances of various internal and external attacks on their data. Sometimes there can be forgeability by cloud server for saving memory space by deleting the data from storage which has not been requested since long time by users. Thus, how efficiently the client can check data authenticity without original data is a big issue. Before storing any sensitive data on remote storage it has to encrypt by using one of the encryption algorithm for securing it. But the traditional methods of accessing data to verify its integrity will not be an efficient approach and also it will not give public auditability. So to achieve confidentiality and authenticity of the data located on remote storage, homomorphic encryption schemes are useful.

Homomorphism gives flexibility of performing operations on the cipher text which is equivalent to work with the original text.

II. LITERATURE SURVEY

In order to ensure or verify whether server exactly possess the same data there was introduction of provable data possession protocol[1].This protocol checks any requested subfield and relies on the proofs provided by storage server, here the auditor maintains tags for checking integrity. The authentication of data is necessary for the data stored on remote storages. To achieve confidentiality and data authentication of sensitive data with better performance in the distributed architecture proper authentication mechanism is required [2].

First time Ateniese et al. [3] have provided a framework for the data which tends to change regularly. In this design, tag verifier uses public key of respective client.

Message authentication code calculation for verifying data is not suitable in the public domain so homomorphic linear authenticators are generated for data authenticity. In the protocol all the metadata's were combined together so it was not possible to verify individual parts of the data [4].

Homomorphic encryption permits operations on cipher text directly, and thus reducing communication and computation cost. Cryptography is tool to provide security to the remotely stored data where data will be always encrypted prior to upload at local platform [5].

Data integrity can get compromised by malicious or non-malicious threats. Better authentication mechanism helps ensuring integrity of the data. Due to various vulnerabilities, auditing of sensitive data is the prime concern. To develop an effective auditing system two essential components are required, web platform and protocol analysis. Network packet data processing by traditional way was causing a huge overhead.

Hence there is need of effective audit system to ensure integrity of remotely store data [6]. In the Merkle Hash Tree technique, signatures are verified for checking data intactness. In this method hash tree root value is required to share by some secure mechanism from client to auditor. For authenticity of data of the any specific sub block only respective hash can be shared. Here in tree structure likewise, particular nodes tags can be shared to verify complete content intactness [7].

Homomorphic authenticator and random masking techniques have been used by authors so that auditor should not get any information of the original contents in auditing process [8].

In case of MAC based techniques data regarding to each block in terms of hash need to be stored so, storage requirement will be of double. The size of the metadata calculated in the RSA-homomorphic implementation will be equal to the size of RSA modulus [9]. To reduce the storage requirement short signatures can be preferred.

III. PROPOSED WORK

Public auditability will not be possible if we use symmetric key cryptosystem. Here, in the system, data owner can use one of the homomorphic algorithm mentioned below to store the data in the encrypted format so can ensure confidentiality of data.

Revised Manuscript Received on January 05, 2020

Vidyullata Devmane*, PhD Scholar of Computer Engineering Department, PAHERU Udaipur, Rajasthan, India.

Dr. B.K.Lande, Professor in Electronics Department, Datta Meghe College of Engineering, Airoli, Navi Mumbai.India.

Dr. Jyoti Joglekar, department, Professor in Computer Engineering Departement,K.J.Somaiya College of Engineering,Mumbai, India

Dr.Hemant Kothari,Professor at PAHERU,Udaipur, Rajasthan,India.

Homomorphic property allows authentic user or data auditor to verify authenticity of the data by performing some operations on cipher text only, which gives the benefit of confidentiality of the data. Verifier will get validation whether the file is intact or not to ensure integrity of remotely stored data [10].

Here we have applied hash tree concept and also MD5 & HMAC MD5 to verify integrity of random blocks instead of whole file. And for achieving confidentiality of data homomorphic algorithms RSA, Paillier, ElGamal are implemented and analyzed.

There are different entities involved in the framework. Before uploading data, owner will calculate message digest and perform encryption by using homomorphic algorithm. The cipher text will be stored on remote storage and hash will be kept with auditor.The file to be uploaded will be converted into numbers for encryption. Owner can also split the file into blocks for using in the hash tree. Here it is possible to decide block size depends upon original file size.

Authentication of data can be done by verifying integrity of block of file using Hash Tree data integrity technique.

Hash Tree is dividing original data into smaller parts and the root hash will reflect overall hash of the data which avoids the unnecessary calculations. While verifying just browse sub file on which integrity check is to be performed not a complete file. Once hash tree is been generated it will be shared with auditor & then he can validate integrity of block of file by specifying the split number so that even if the file name is changed, owner can verify the integrity of that block.

Encrypted data uploaded by authentic owner will be managed securely by cloud storage server which sends the data to the owner, authentic users & auditor as per requests.

Only authentic users can get the data which stored at storage and also can request for data validation to auditor.

Auditor can work on behalf of storage server by performing integrity check of the data by maintaining confidentiality.

IV. EXPERIMENTAL RESULTS

Implementation is done in Java 7.0 language on Windows 7 and CPU speed 1.2 GHz or above with 2 GB RAM. Data is uploaded after encryption on openshift open source cloud storage.Basically time to compute encryption and decryption is observed. To make the algorithms suitable for practical implementation original contents are divided into sub files. While checking time performance of different size files like 15, 128, 352, 512, 1024 bytes etc.are considered. The tables 1 to 3 are providing details of performance of RSA,Paillier and ElGamal algorithms for the sub files after dividing the original larger files. Only two file sizes are mentioned in the tables with time required for computations in encryption and decryption algorithms. Key size for RSA is 1024 bits size and for Paillier and ElGamal its 512 bits size.

Table- 1: Analysis for RSA algorithm with sub files

Size of Sub File (Bytes) / Key Size(Bits)	Encrypted File Size	Time for Encryption	Time for Decryption
352/1024	13.2 kb	0.436 sec	0.653 sec
15/1024	617 bytes	0.016 sec	0.094 sec

Table- 2: Analysis for ElGamal algorithm with sub files

Size of Sub File (Bytes) /Key Size(Bits)	Encrypted File Size	Time for Encryption	Time for Decryption
352 /512	12.9 kb	0.203 sec	0.294 sec
15 /512	595bytes	0.032sec	0.016sec

Table-3: Analysis for Paillier algorithm with sub files

Size of Sub File (Bytes) /Key Size(Bits)	Encrypted File Size	Time for Encryption	Time for Decryption
352 /512	6.39 kb	0.094 sec	0.249 sec
15 /512	309 bytes	0.011 sec	0.057sec

Here encrypted file sizes in the RSA and ElGamal algorithms are approximately same. Time required to encrypt and decrypt the files are having some differences. Even with smaller key size ElGamal is providing good results as compared to RSA & can give better security as it is mathematically stronger. Paillier is generating smaller file size as compared to both RSA & ElGamal algorithms. Also it is mathematically stronger against different attacks and takes lesser time for encryption. As we will increase size file size, time to perform encryption and decryption will be reduced.

We have also implemented hash tree scheme where different blocks will be stored in the various files.

Total number of blocks can be decided depends on size of original data. Procedure is explained for hash tree nodes modification and verification for making the protocol suitable to work with frequently changing data by avoiding complete file access.

n=size of file

p=number of blocks in the last level

Height of the tree=p/2

i=node number in the respective level which starts with zero

After modification of any specific node there is need of reading the hash of the other node to be concatenated with hash of the input node.

If (i mod2==0)

j=i+1

else j=i-1

Concatenate i&j numbered node contents of the same level.

Finding the concatenated hash to be compared in the one level above.

k=int[i div 2]

Compare the k node contents which are already stored.

If original file size is very big and some part may tends to change frequently where most of the contents are static then one can consider hash tree technique. All the different blocks of hash are stored at last level of the tree and every two nodes are concatenated together. Likewise root of the tree will provide overall hash value. Initially complete tree needs to be prepared and afterwards only changing contents will be treated.

V. CONCLUSION

Client's encrypted file is operated directly using homomorphic schemes so reduces computational cost. The Paillier algorithm which is a probabilistic cryptosystem is very strong against various attacks as it will not reveal any clues about original text. Comparative analysis proves that Paillier is much better than ElGamal & RSA in terms of space and time complexity. It would allow any authentic user to perform integrity check. Storing the data with a strong encryption method will ensure authentication of data. Due to hash tree concept, if any sub file will get changed by attacker then the root will not match. However, these data integrity checking schemes uses the existing algorithms like MD5, SHA256 etc. to calculate message digest so we can change hash algorithm as per size of the hash requirement. This work ensures the privacy of user's data while checking integrity.

REFERENCES

1. Jingwei Li, Jin Li, Dongqing Xie and Zhang CAI, "Secure Auditing and Deduplicating Data in Cloud", DOI 10.1109/TC.2015.2389960, IEEE Transactions on Computers.
2. Nedhal A. Al-Sayid, Dana Aldlaeen, "Database Security Threats: A Survey Study".2013 5th International Conference on Computer Science and Information Technology (CSIT).
3. Ateniese, G. Karmara,S.Katz, "Proofs of storage from homomorphic
4. identification protocols", Proceedings of 15th international conference on the theory and application of cryptography and Information Security: Advances in Cryptology, ASIACRYPT '09, pp. 319–333. Springer, Berlin, Heidelberg (2009).
5. KeheWu, Liang Hua, Xiaoxiang Wang & Xuewei Ding, "The Design and Implementation of Database Audit System Framework", 978-1-4799-3279-5 /14/$31.00 ©2014 IEEE
6. Giuseppe Ateniese, Randa Burns,Reza Curtmola, Joseph Herring,Lea Kissner, Zachary Peterson & Dawn Song, "Provable Data Possession at Untrusted Stores", ACM Conference on Computer and Communications Security. ACM, 2007.
7. Jingwei Li, Jin Li, Dongqing Xie and Zhang Cai, "Secure Auditing and De duplicating Data in Cloud", DOI 10.1109/TC.2015.2389960, IEEE Transactions on Computers
8. Muhammad Saqib Niaz, Gunter Saake, "Hash Tree based Techniques for Data Integrity of Outsourced Data", 27th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 26.05.2015-29.05.2015, Magdeburg, Germany.
9. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing", In: InfoCom2010, IEEE, March 2010.
10. Kan Yang ,Xiaohua Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities". Springer Science+Business Media, LLC 2011.
11. Snehal Sawant, Vidyullata Devmane. "Towards privacypreserving for dynamic data in cloud storage", 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS),2017.

AUTHORS PROFILE

Vidyullata Devmane, received B.E. Computer Engineering degree from Walchand College of Engineering ,Maharashtra, India. She possesses Masters degree in Computer Engineering from the University of Mumbai, India. And she is persuing Ph.D in Computer Engineering from PAHERU, Rajasthan, India. She is currently working as an Associate Professor with the Department of Computer Engineering, Shah & Anchor Kutchhi Engineering College, Mumbai. Her areas of interests are Cryptography & System Security, Advanced Algorithms and System Programming.

B.K.Lande has received the Ph.D degree from the Indian Institute of Technology (IIT) Bombay, Masters degree from Walchand College of Engineering, Maharashtra, and B.E. degree from Nagpur University, Maharashtra, India. Currently he is working as a Professor in Datta Meghe College of Engineering, Airoli, Navi Mumbai. His areas of research are Controls & Communication, Image Processing and Cryptography & System Security. Reviewer of various reputed publications.

Jyoti Joglekar received the Ph.D degree from the Indian Institute of Technology (IIT) Bombay, Mumbai, India, in 2015. She possesses Master of Engineering degree in computer engineering from the University of Mumbai, Mumbai, India. She is currently a Professor with the Department of Computer Engineering, K. J. Somaiya College of Engineering, Mumbai, India. Her areas of research are satellite image processing and analysis, machine learning and big data analytics, Cryptography & System Security. She is a Life Member of the Indian Society for Remote Sensing and a Fellow member of IETE. Reviewer of various reputed publications.

Hemant Kothari, is currently working as Professor at PAHERU, Udaipur. And holding a post of Dean PG at Pacific Academy of Higher Education and Research University, Udaipur, Rajasthan, India. He has twenty years of teaching experience, Ten students have completed PhD successfully under his guidance. He is Reviewer of various reputed publications.