International Journal of Recent Trends in Engineering & Research (IJRTER)
Conference on Recent Trends in Electrical and Electronics Engineering (RTEEE '19)
Special Issue; March- 2019 [ISSN: 2455-1457]
DOI: 10.23883/IJRTER.CONF.20190304.030.TASBC

Preserving Remote Data Integrity from Spoofing Attack in Cloud Environment using Probabilistic Checking Method

Dr.T. Avudiappan M.E,PhD.,
Assistant Professor
Department of Computer Science and Engineering
K.Ramakrishnan College of Technology
Samayapuram,Trichy,India

R.Priya
P.G Scholar
Department of Computer Science and Engineering
K.Ramakrishnan College of Technology
Samayapuram,Trichy, India

Abstract- **Remote data integrity checking is a crucial technology in cloud computing. Recently many works focus on providing data dynamics and/or public verifiability to this type of protocols. Existing protocols can support both features with the help of a third party auditor. In existing system, they propose a remote data integrity checking protocol that supports data dynamics. This paper adapts existing protocol to support public verifiability. The proposed protocol supports public verifiability without help of a third party auditor. In addition, the proposed protocol does not leak any private information to third party verifiers. Through a formal analysis, we show the correctness and security of the protocol. After that, through theoretical analysis and experimental results, we demonstrate that the proposed protocol has a good performance.**

Keywords— Data Integrity, data dynamics, Public verifiability, Privacy.

I INTRODUCTION

Many cloud storage providers declare that they store multiple replicas of clients' data in order to prevent data loss. However, currently there is no guarantee that they actually spend storage for multiple replicas. Recently a multiple-replica provable data possession (MR-PDP) protocol is proposed, which provides clients with the ability to check whether multiple replicas are really stored at the cloud storage servers. However, in MR-PDP, only private verifiability is achieved. In this project, we propose a multiple-replica remote data possession checking protocol which has public verifiability. The public verifiability increases the protocol's flexibility in that a third-party auditor can perform the data checking on behalf of the clients. Homomorphic authentication tags based on BLS signature are used in the proposed protocol. By security analysis and performance analysis, the proposed protocol is shown to be secure and efficient, which makes it very suitable in cloud storage systems.

Remote Integrity checking protocol

In remote data integrity checking protocols, the client can challenge the server about the integrity of a certain data file, and the server generates responses proving that it has access to the complete and uncorrupted data. The basic requirements are that the client does not need to access the complete original data file when performing the verification of data integrity, and that the client should be able to verify integrity for an unlimited number of times. Furthermore, the protocol needs to be secure against a malicious server that tries to pass the data integrity verification without access to the complete and uncorrupted data.

Most previously proposed protocols meet all the above basic requirements. Nevertheless, in addition to these requirements, there are also three advanced requirements: data dynamics, public verifiability and privacy against verifiers. In the following we introduce the motivations and definitions of these advanced requirements. In a realistic application, these advanced features may be needed at the same time. For example, consider an online document system, in which the client can create and modify her documents. The client can also cooperate on a document with her partners. The remote data integrity checking can ensure the integrity of the client's documents. When the client

or her partners modify the document, the document and the tags need to be updated. This is supported by data dynamics. However, if the file gets corrupted and the server refuses to admit it, then a third party authority can easily prove the data corruption to the court. This is supported by the public verifiability. Furthermore, if the document contains sensitive data, such as business secrets, the client will not want it to be disclosed to the third party authorities. The privacy against third-party verifiers can ensure that. The previously proposed protocols try to achieve these advanced features. The protocols in support data dynamics at the block level, including block insertion, block modification and block deletion. The protocol of [4] supports data append operation. In addition, the protocol in [1] can be easily extended to support data dynamics.

A remote data integrity checking protocol for cloud storage, which can be viewed as an adaptation of Sebe et al.'s protocol . The proposed protocol inherits the support of data dynamics from , and supports public verifiability and privacy against third party verifiers, while at the same time it doesn't need to use a third-party auditor.

To give a security analysis of the proposed protocol, which shows that it is secure against the untrusted server and private against third party verifiers.

II RELATED WORKS

Checking data possession in networked information systems[1] such as those related to critical infrastructures is a matter of crucial importance. Remote data possession checking protocols[10] permit checking that a remote server can access an uncorrupted file in such a way that the verifier does not need to know beforehand the entire file that is being verified. Unfortunately, current protocols only allow a limited number of successive verifications or are impractical from the computational point of view. In this work, presents a new remote data possession checking protocol such that. a) It allows an unlimited number of file integrity verifications. b) Its maximum running time can be chosen at set-up time and traded off against storage at the verifier.

[5]Presented a dynamic replicated data possession checking scheme for checking the data copies integrity in the Cloud Service Provider. The data owners encrypt their data using FHE algorithm, obtain the data replicas, and then store their encryption data replicas on the CSP. Firstly, the proposed scheme not only supports checking integrity of the data file copies anytime and anywhere but also satisfies public verification without leaking any information to the third-party auditor. Secondly, the authorized users can decrypt data copies received from the CSP using a single decryption secret key. Thirdly, the data owner can implement data block update operation on data replicas. Finally, the retrieval algorithms in this scheme can identify the corrupted data blocks in corrupted copies accurately. The corrupted data block copies can be reconstructed by the data owner, and thus he/she needs not to generate the whole corrupted copy file. Through security analysis, they have shown that our scheme can resist forgery attacks, replacement attack, and replay attack. At the same time, our DPRDP scheme achieves a higher efficiency than DMR-PDP scheme.

PROVABLE DATA POSSESSION AT UNTRUSTED STORES

Introduce a model for Provable Data Possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original

DOI: 10.23883/IJRTER.CONF.20190304.030.TASBC

data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage systems. We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.

MULITPLE - REPLICA PROVABLE DATA POSSESSION

Many storage systems rely on replication to increase the availability and durability of data on untrusted storage systems. At present, such storage systems provide no strong evidence that multiple copies of the data are actually stored. Storage servers can collude to make it look like they are storing many copies of the data, whereas in reality they only store a single copy. They address this shortcoming through Multiple-Replica Provable Data Possession (MR-PDP): A provably-secure scheme that allows a client that stores replicas of a file in a storage system to verify through a challenge-response protocol that each unique replica can be produced at the time of the challenge and that the storage system uses times the storage required to store a single replica. MR-PDP extends previous work on data possession proofs for a single copy of a file in a client/server storage system. Using MR-PDP to store t replicas is computationally much more efficient than using a single-replica PDP scheme to store t separate, unrelated files. Another advantage of MR-PDP is that it can generate further replicas on demand, at little expense, when some of the existing replicas data. However, distributed storage systems lack constructs that allow them to securely determine the number and location of replicas in the system. Distributed storage systems that perform replica maintenance often have storage sites crosscheck the contents of replicas through content hashing. Recently, there has been much interest in having clients check that servers have a copy of the data. Both types of protocols are vulnerable to collusion attacks in which multiple servers that appear to be storing multiple replicas are in fact storing only a single copy of the data. In general, this can be done by redirecting and forwarding challenges from the multiple sites to the single site that stores the data.

PROOFS OF RETRIEVABILITY VIA HARDNESS AMPLIFICATION

Many organizations and even average computer users generate huge quantities of electronic data. Although advances in hard-disk capacity have mostly kept up, allowing most users to store their

data locally, there are many reasons not to do so. Users worried about reliability want to have replicated copies of their files stored remotely in case their local storage fails. Remotely stored data can be made accessible from many locations making it more convenient for many users. Some companies provide useful functionality on remotely stored data using the "software as a service" model. For example, many web-based e-mail services provide tools for searching and managing remotely stored e-mails, making it beneficial for users to store these remotely. Lastly, some organizations create large data sets that must be archived for many years, but are rarely accessed and so there is little reason to store such data locally.

Proofs of Retrievability.

One problem with remote storage is that of accountability. If remotely stored data is rarely accessed, how can users be sure that it is being stored honestly? For example, if a remote storage provider experiences hardware failure and loses some data, it might reason that there is no need to notify its clients, since there is a good chance that the data will never be accessed and, hence, the client would never find out! Alternatively, a malicious storage provider might even choose to delete rarely accessed files to save money. To assuage such concerns, we would like a simple auditing procedure for clients to verify that their data is stored correctly. Such audits, called Proofs of Retrievability (PoR), were first formalized by Juels and Kaliski in [JK07]. In a PoR protocol a client stores a file F on a server an keeps only a very short private verification string locally. Later, the client can run an audit protocol in which it acts as a verifier while the server proves that it possesses the client‟s data. The security of a PoR protocol is formalized by the existence of an extractor that retrieves the original file F from any adversarial server that can pass an audit with some reasonable probability. One simple PoR protocol would be for the client to sign the file F and store only the verification key locally. Then, to run an audit, the server would send the file along with the signature. Of course, for practical use, we are interested in schemes with significantly better efficiency.

III SYSTEM DESIGN

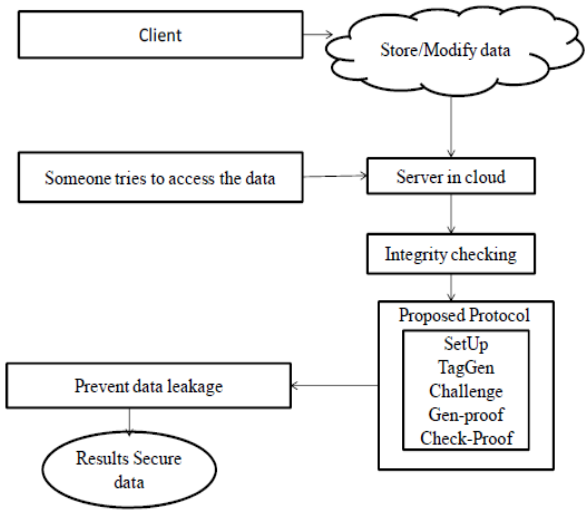

Fig 1 System architecture

Figure 1 shows the system architecture for the proposed system. The data dynamics phase allows Client store their data to remote server via cloud and dynamically updates their data like

insertion, modification and deletion. Public verifiability module performs integrity checking operation. This can be done by third party verifier. Third party verifier verify client's data whether it is original or not.

Propose remote data integrity checking protocol is carried out. When the verification is performed by a third party verifier (not by the client) the protocol must ensure that no private information contained in the data is leaked.

IV SYSTEM WORKING

REMOTE FILE INTEGRITY CHECKING PROTOCOL

Section I describe the proposed remote data integrity checking protocol. Just as mentioned in Section II, the proposed protocol has functions SetUp, TagGen, Challenge, GenProof and CheckProof, as well as functions for data dynamics. In the following we present the former five functions of the proposed protocol.

SetUp(1k) → (pk, sk): Let N = pq be one publicly known RSA modulus, in which p = 2p ′ + 1, q = 2q ′ + 1 are two large primes. p ′ and q ′ are also primes. In addition, all the quadratic residues modulo N form a multiplicative cyclic group, which we denote by QRN . Denote the generator of QRN by g. 1 Since the order of QRN is p ′ q ′ , the order of g is also p ′ q ′ . Let pk = (N, g) and sk = (p, q). pk is then released to be publicly known to everyone, and sk is kept secret by the client.

TagGen (pk, sk, m) → Dm: For each file block mi , i ∈ [1, n], the client computes the block tag as Di = (g mi) mod N.

Without loss of generality, we assume that each block is unique. If in some particular applications, there exist blocks with the same value, then we differentiate them by adding a random number in each of them. Let Dm = {D1, D2, ..., Dn}. After finishing computing all the block tags, the client sends the file m to the remote server, and releases Dm to be publicly known to everyone.

Note that in the TagGen function, we make all the blocks distinct by adding random numbers in blocks with the same value. If the server still tries to save its storage space, then the only way is by breaking the prime factorization of N, or equally, getting a multiple of ϕ(N). The hardness of breaking large number factorization makes the proposed protocol secure against the untrusted server.

SECURE HASH FUNCTION

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. For a hash function for which L is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in approximately 2L evaluations. This is called a pre image attack and may or may not be practical depending on L and the particular computing environment. However, a collision, consisting of finding two different messages that produce the same message digest, requires on average only about 1.2 × 2L/2 evaluations using a birthday attack. Thus the strength of a hash function is usually compared to a symmetric cipher of half the message digest length. SHA-1, which has a 160-bit message digest, was originally thought to have 80-bit strength.

V CONCLUSION

This system proposes a new integrity checking protocol for cloud storage. The proposed protocol is suitable for providing integrity protection of customers' important data. The proposed protocol supports data insertion, modification and deletion at the block level, and also supports public verifiability. The proposed protocol is proved to be secure against an untrusted server. It is also private against third party verifiers. The proposed protocol will be effective for communication and storage cost in cloud storage. Currently this paper focuses on working of extending the protocol to support data level dynamics. The difficulty is that there is no clear mapping relationship between the data and the tags. In the current construction, data level dynamics can be supported by using block level dynamics. Whenever a piece of data is modified, the corresponding blocks and tags are updated. However, this can bring unnecessary computation and communication costs. We aim to achieve data level dynamics at minimal costs in our future work

References

[1] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, J.-J. Quisquater, (2008) "Efficient Remote Data Possession Checking in Critical Information Infrastructures", IEEE Trans. Knowledge and Data Eng., no. 8, pp. 1034-1038, Aug.

[2] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, (2009) "Cloud Computing and Emerging IT Platforms: Vision Hype and Reality for Delivering Computing as the Fifth Utility", Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616,

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, D. Song, (2007) "Provable Data Possession at Untrusted Stores", Proc. 14th ACM Conf. Computer and Comm Security (CCS '07), pp. 598-609

[4] R. Curtmola, O. Khan, R. Burns, G. Ateniese, "MR-PDP: (2008) Multiple-Replica Provable Data Possession", Proc. 28th Int'l Conf. Distributed Computing Systems (ICDCS '08),

[5] C. Erway, A. Kp, C. Papamanthou, R. Tamassia, (2009) "Dynamic Provable Data Possession", Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), pp. 213- 222,

[6] C. Wang, Q. Wang, K. Ren, W. Lou, (2009) "Ensuring Data Storage Security in Cloud Computing", Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), pp. 1-9, [7] D.L.G. Filho, P.S.L.M. Barreto, (2006) "Demonstrating Data Possession and Uncheatable Data Transfer."

[8] M.A. Shah, M. Baker, J.C. Mogul, R. Swaminathan, (2007)"Auditing to Keep Online Storage Services Honest", Proc. 11th USENIX Workshop Hot Topics in Operating Systems (HOTOS).

[9] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, S.S. Yau,(2010) "Cooperative Provable Data Possession".

[10] Z. Hao, N. Yu, "A Multiple-Replica Remote Data Possession Checking Protocol with Public Verifiability", (2010) Proc.Second Int'l Data Privacy and E-Commerce Symp.(ISDPE '10)

[11] K. D. Bowers, A. Juels, and A. Oprea, (2009) "HAIL: a high-availability and integrity layer for cloud storage," in CCS '09: Proceedings of the 16th ACM conference on Computer and

Special Issue; March- 2019 [ISSN: 2455-1457]
DOI: 10.23883/IJRTER.CONF.20190304.030.TASBC

communications security,(New York, NY, USA), pp. 187–198, ACM,
[12] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in TCC '09: Proceedings of the 6th Theory of Cryptography Conferenceon Theory of Cryptography, (Berlin,Heidelberg), pp. 109–127, Springer-Verlag, 2009.

