Asian Journal of Electrical Sciences (AJES)
Vol.3.No.1 2015 pp 42-45
available at: www.goniv.com
Paper Received :08-03-2015
Paper Accepted:20-03-2015
Paper Reviewed by: 1. R. Venkatakrishnan 2. R. Marimuthu
Editor : Prof. P.Muthukumar

PRESERVING THE CONFIDENTIALITY AND SECURITY FOR THE DATA STORED IN CLOUD

[1]K. ANITHA, [2]S. ISWARYA, [3]K. VENILLA,
Final Year B.E. – CSE, Sri Aravindar Engineering College,
[1]anithakarunanithi03@gmail.com, [3]yajanila94@gmail.com

ABSTRACT

The main aim of this project is to preserve privacy from the third party auditors. For preserving privacy I am going to explore a context in this project. ***Remote data integrity checking is one of the important technologies in cloud computing. Now a day's many works focus on providing data dynamics and/or public verifiability to this type of protocols. In existing system, it supports both features with the help of a third party auditor. In previous work, Sebe'tal proposes a remote data integrity checking protocol that supports data dynamics. In this project, I adapt Sebe`tal's protocol to support public verifiability and I am going to implement the public verifiability without the help of any third party auditors. In addition, the proposed protocol produces high security than the previous system and preserves the confidentiality from the third party verifiers. Here I am going to use RSA and HMAC algorithm for compressing and decompressing the message in order to preserve the security and confidentiality.*** **The data stored in the cloud is so important that the clients must ensure it is not lost or corrupted.** ***Thus by means of a formal analysis, I show the correctness and security of the protocol. After that, through theoretical analysis and experimental results, I demonstrate that the proposed protocol has a good performance.***

Key Terms — Data integrity, Data dynamics, Public verifiability, Privacy.

I . INTRODUCTION

Storing data in the cloud has become a modern trend. An increasing number of clients store their important data in remote servers in the cloud, without leaving a copy in their local computers.
In this paper, we have the following main contributions:

- We propose a remote data integrity checking protocol for cloud storage, which can be viewed as an adaptation of Sebe'etal's protocol [1]. The proposed protocol inherits the support of data dynamics from [1], 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.
- We give a security analysis of the proposed protocol, which shows that it is secure against the untrusted server and private against third party verifiers.
- We have theoretically analyzed and experimentally tested the efficiency of the protocol. Both theoretical and experimental results demonstrate that our protocol is efficient.

Sometimes the data stored in the cloud is so important that the clients must ensure it is not lost or corrupted *[2]*. While it is easy to check data integrity after completely downloading the data to be checked, downloading large amounts of data just for checking data integrity is a waste of communication bandwidth. Hence, a lot of works have been done on designing remote data integrity checking protocols, which allow data integrity to be checked without completely downloading the data.

Remote data integrity checking is first introduced in which independently propose RSA based methods for solving this problem. After that Sebe'et al propose a remote storage auditing method based on pre-computed challenge-response pairs. Recently many works focus on providing three advanced features for remote data integrity checking protocols: data dynamics, public verifiability and privacy against verifiers. The protocols support data dynamics at the block level, including block insertion, block modification and block deletion. The protocol supports data append operation.

II .PROBLEM DEFINITION

In existing protocols data dynamics at the block level, including block insertion, block modification and block deletion. Protocols support public verifiability, by which

anyone can perform the integrity checking operation and also support privacy against third party verifiers.
We need to design a remote data integrity checking protocol that includes the following five functions:

SetUp, TagGen, Challenge, GenProof and CheckProof.

SetUp(1^k) → (*pk, sk*): Given the security parameter k, this function generates the public key *pk* and the secret key *sk*. *pk* is the public to everyone, while *sk* is kept secret by the client.

TagGen(*pk, sk, m*) → D_m : Given *pk, sk* and *m*, this function computes a verification tag D_m and makes it publicly known to everyone. This tag will be used for public verification of data integrity.

Challenge(*pk*, D_m) → chal: Using this function, the verifier generates a challenge chal to request for the integrity proof of file *m*. The verifier sends chal to the server.

GenProof(*pk*, D_m , *m*, chal) → *R*: Using this function, the server computes a response *R* to the challenge chal. The server sends *R* back to the verifier.

CheckProof (*pk*, D_m , chal, *R*) → {"success", "failure"}: The verifier checks the validity of the response *R*. If it is valid, the function outputs **"success"**, otherwise the function outputs **"failure"**. The secret key *sk* is not needed in the CheckProof function.

There are two security requirements for data integrity checking protocol:

- Security against the server with public verifiability.
- Privacy against the third party verifiers.

III . PROPOSED SYSTEM

I propose a remote data integrity checking protocol for cloud storage, which can be viewed as an adaptation of Sebe` et al.'s protocol*[5]*. The proposed protocol inherits the support of data dynamics, 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. I give a security analysis of the proposed protocol, which shows that it is secure against the untrusted server and private against third party verifiers.

In this section, protocol supports data dynamics at the block level. In following we show how our protocol supports block modification. In our proposed system we are going to implement RSA algorithm for generating Random ID for the users which is useful to avoid hacking from others.

Also we are going to implement the HMAC algorithm for supporting the above said data dynamics concept in well secured manner whish gives high confidentiality, authentication and privacy from third party verifiers*[7]*.

IV . IMPLEMENTATION OF HMAC

A. MAC

MAC stands for Message Authentication Code. It's basically a checksum for data going though in secure channel.

When using MAC, two parties, e.g. Alice and Bob need to share a secret key K, and agree with some MAC algorithm in the first place. If Alice sends a message M to a Bob, Alice first passes the message and the shared secret key K into the MAC algorithm, thus to generate a MAC code MAC (M, K). Alice then sends Bob the message M along with the MAC (M, K). After receiving M and MAC (M, K), Bob generates his own MAC code on top of the message M he received plus the shared secret key K (using the same MAC algorithm), and verifies that the MAC code he generated matches the one sent by Alice.

A general step-by-step process of how a generic MAC function works can be described as following:

1. *Sender sends Message & MAC (Message, K), M1.*
2. *Receiver receives both parts.*
3. *Receiver makes his own MAC (Message, K), M2.*
4. *If M2! = M1, data has been corrupted.*
5. *If M2 == M1, data is valid.*

B. HMAC

HMAC stands for Hash-based MAC. It works by using an underlying hash function over a message and a key.

HMAC generates a Message Authentication Code by the following formula:

HMAC(M) = H[(K+opad) & H[(k+ipad) & M]]

M = Message
H[] = Underlying Hash function
K = Shared Secret Key
opad = 36hex, repeated as needed
ipad = 5Chex, repeated as needed
& = concatenation operation
+ = XOR operation

The HMAC (M) is then sent as any typical MAC(M) in a message transaction over insecure channels (See section 1). Again, any hash function can be used, but MD5 and SHA-1 seem to be most popular.

C. Uses of HMAC

Speed is the main reason. Hash functions are much faster than block ciphers such as DES and AES in software implementation. However, HMAC, as a cryptographic mechanism, is repudiatable. That is, Bob cannot demonstrate that data really came from Alice -- both a sender and a receiver can generate an exactly same HMAC output (so Bob could have made the data

himself). This is unlike digital signatures which only the sender can generate.

D. System Architecture

The above figure 1 shows the architecture diagram of the proposed system. It performs five different processes as shown below.

i. Create Request
ii. Random ID Generation
iii. Send Data to Cloud
iv. Send Patient Data Request
v. Received Patient Details

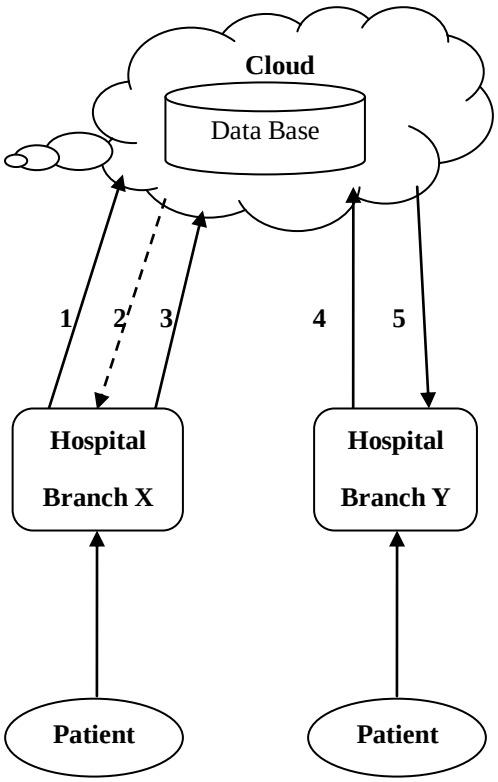

Fig 1. System Architecture

E. Module Descriptions

The proposed system of my project consists of 5 modules. They are

- Login
- Cloud Random Word Generation
- Treatment
- Branches Controls
- Security

i. Login

In this module I designed the Login screen where user or authenticated hospital branch can get user id, password from cloud. If they have created user ID, they can update data to cloud. If new user wants to get access from cloud they must register about the branch. While registering the client, server will provide the user id and that id will be unique for each user.

ii. Random Word Generation

In this module I have designed the server which randomly generate the words, at the same time all the other processes also handling in this module. The processes of the server are update the word to each patient, register the patient details, add the server entry in routing table, generate the word for next entity, the KEA1-r and the large integer factorization assumptions, the proposed protocol is secure against the untrusted server*[3]*.

iii. Treatment

After getting the Random word the branches can update all data to cloud under the random id, the doctor‘s are going to update all treatment under the random id, if the patient goes any other same hospital branch they can easily get treatment through random id, the hospital doctor can get data from cloud without third person verification.

iv. Branches Control

All branches don’t control cloud. The cloud is control all branches, In this module I designed the server which randomly generate the words, at the same time all the other processes also handling in this module. After getting the Random word the branches can update all data to cloud under the random id, the doctor‘s are going to update all treatment under the random id, if the patient goes any other same hospital branch they can easily get treatment through random id, the hospital doctor can get data from cloud without third person verification*[4]*.

v. Security

All patients are getting new random id but the large integer factorization assumptions, the proposed protocol is secure against the UN trusted server, literature. The main issue is how to frequently, efficiently and securely verify that a storage server is faithfully storing its client's (potentially very large) outsourced data. The storage server is assumed to be untrusted in terms of both security and reliability.

V . CORRECTNESS AND SECURITY ANALYSIS

In this section, we first show that the proposed protocol is correct in the sense that the server can pass the verification of data integrity as long as both the client and the server are honest. Then we show that the protocol is secure against the untrusted server. These two theorems together guarantee that, assuming the client is honest, if and only if the server has access to the complete and uncorrupted data, it can pass the verification process successfully. Finally we show that the proposed protocol is private against third party verifiers*[9]*.

Theorem 1: If both the client and the server are honest, then the server can pass the verification successfully.

Theorem 2: Under the KEA1-r and the large integer factorization assumptions, the proposed protocol is secure against the untrusted server.

KEA1-r (Knowledge of Exponent Assumption):

For an adversary A taking input (N, g, g^s) and (C,Y) with Y = C^s, there exists "extractor" which given the same input as A return c such that C = g^c.

Theorem 3: (Privacy against Third Party Verifiers)Under the semi-honest model, a third party verifier cannot get any information about the client's data m from the protocol.

VI . DATA DYNAMICS

The proposed protocol supports data dynamics at the block level in the same way as [1]. In the following we show how our protocol supports block modification. Due to space limitation, we describe the support of block insertion and block deletion in the full version.

Block Modification: Assume that the client wants to modify the *ith* block *mi* of her file. Denote the modified data block *m**. Then the server updates *mi* to *m**. Next client computes a new block tag for the updated block, i.e., $D^* = g^{m^*} \bmod N$.

From the above we can see that the correspondence relationship between the block and the digest does not change after the data updating, i.e., $D_i = g^{mi} \bmod N$, $i = 1, 2, ..., x|m|/l$. So the data integrity is still protected. If the client wants to make sure that the file has really been updated, she can launch a proof request immediately by sending a challenge to the server. Any block that is updated is given a novel random number, so that each block remains unique. Therefore, the server cannot delete any block without being detected.

VII . CONCLUSION AND FUTURE WORK

I propose a lightweight and non-path-based mutual anonymity protocol for P2P systems, Rumor Riding (RR) protocol. Employing a random walk concept, RR issues key rumors and cipher rumors separately and expect that they meet in some random peers. The results of trace-driven simulations and simple implementations show that RR provides a high degree of anonymity and outperforms existing approaches in terms of reducing the traffic overhead and processing latency. I also discuss how RR can effectively defend against various attacks. Future and ongoing work includes accelerating the query speed, introducing mimic traffic to confuse attackers, and optimizing the k and L combination to further reduce the traffic overhead. I will also investigate other security properties of RR, such as the unlinkability, information leakage and failure tolerance when facing different attacks. It would also be interesting to explore the possibility of implementing this lightweight protocol in other distributed systems, such as grid systems and ad-hoc networks. I aim to achieve data level dynamics at minimal costs in future work.

REFERENCES

1. *Ateniese. G, Burns. R, Curtmola. R, Herring. J, Kissner. L, Pe- terson. Z and Song. D. (2007) 'Provable data possession at untrusted stores' in CCS'07, (New York, NY, USA), pp. 598–609.*
2. *Ateniese. G, Di Pietro. R, Mancini. L. V and Tsudik. G. (2008) 'Scalable and efficient provable data possession' in SecureComm'08, ACM.*
3. *Chaum. D. (1981) 'Untraceable Electronic Mail Return Addresses and Digital Pseudonyms' Comm. ACM, Vol. 24, No. 2, pp. 84-90.*
4. *Curtmola. R, Khan. O, Burns. R and Ateniese. G. (2008) 'MR-PDP: Multiple-Replica Provable Data Possession' in ICDCS'08, IEEE.*
5. *Erway. C, Ku pc u. A, Papamanthou. C and Tamassia. R. (2009) 'Dynamic provable data possession' in CCS'09, pp. 213–222, ACM.*
6. *Reiter. M. K. and Rubin. A. D. (1998) 'Crowds: Anonymity for Web Transactions' ACM Trans. Information and System Security, Vol. 1, No. 1, pp. 66-92.*
7. *Rivest. R, Shamir. A, and Adleman. L. (1978) 'A Method for Obtaining Digital Signatures and Public-Key Cryptosystems' Comm. ACM, Vol. 21, No. 2, pp. 120-126.*
8. *Sebe. F, Domingo-Ferrer. J, Martinez-Balleste. A, Deswarte. Y and Quisquater. J. J. (2008) 'Efficient remote data possession checking in critical information infrastructures' IEEE Trans. on Knowledge and data Engineering, Vol. 20, pp. 1034 –1038.*
9. *Sherwood. R, Bhattacharjee. B and Srinivasan. A. (2002) 'P5: A Protocol for Scalable Anonymous Communication' Proc. IEEE Symp. Security and Privacy, pp. 58-70.*
10. *Xiao. L, Xu. Z, and Zhang. X. (2003) 'Low-Cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks' IEEE Trans. Parallel and Distributed Systems, Vol. 14, No. 9, pp. 829-840.*
11. *Zhuo Hao, Sheng Zhong and Nenghai Yu (2011) 'A Privacy – Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability' IEEE Transactions on Knowledge and Data Engineering, vol. 23, no. 9, pp. 1432-1437.*
12. *I. Damgård, "Towards practical public key systems secure against chosen ciphertext attacks," in CRYPTO'91, Springer-Verlag, 1992.*
13. *Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," SUNY Buffalo CSE department technical report 2010-11, 201*