International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 17
November 2016

Storage Auditing Protocol to Prevent Access to Keys

Nalla Swapna
MTech(CSE)
SLC's Institute of Engineering and Technology

Adavelli Ramesh*Mtech*
Assoc. Prof Dep. of computer Science
Email Ramesh.adavalli@gmail.com
SLC's Institute of Engineering and Technology

Manchukonda Kishore*Mtech*
Asst. Prof Dep. of computer Science
Email : Kishore.manchukonda@gmail.com
SLC's Institute of Engineering and Technology.

ABSTRACT

Auditing is an important service provided by cloud, to check the correctness of the date which stored in the public cloud. All existing protocols are unable to work because week sense of security provided at client side. So in order to overcome from this disadvantage we are going to implementing new approach called auditing protocol to prevent access to keys, and also we develop new approach called block less verifiability. Performance study shows that our proposed protocol is safe and capable.
Keywords: Public cloud, storage auditing, Key exposure conflict.

1.INTRODUCTION:

Cloud computing is an isolated servers hosted on the internet to store, manage and process data moderately than a local server or a personal computer. Auditing protocol is used to check correctness of the data stored in the public cloud. In order to overcome from computation overhead and communication overhead we are using Homomorphic Linear Authenticator (HLA) technique [1]- [5].In order to overcome from computational burden we are using third party auditors (TPA) [1]-[5]. To check the correctness of data stored TPA's are used.

Auditing protocols in [2] and [13] also supports for dynamic operations. Many research works about auditing include in recent years, a critical security problem-the key disclosure problem is undefined.

The client's secrete key is exposed due to several reasons. Firstly, Key management is very complex procedure which includes many factors including system policy, user training etc...Some time client going choose cheep software based key management for economical factor, which provides limited security so secrete keys explored to others. Sometimes client himself may be target and vulnerable to many internet based security attacks, for ordinary clients the sense of security is weaker.

2.EXISTING SYSTEM:

Two basic solutions for the key-disclosure problem for cloud storage auditing.

A. Naive solution:

In naïve solution the client uses the traditional key revocation method. Once client knows his secret key is exposed, he will revoke this secret key and the corresponding public key. Meanwhile, he generates one new pair of secret key and public key, and publishes the new public key by the certificate update. The authenticators of the data previously stored in cloud, however, all need to be updated because the old secret key is no longer secure. Thus, the client needs to download all his previously stored data from the cloud, produce new authenticators for them using the new secret key, and then upload these new authenticators to the cloud. It is a complex procedure, and consumes a lot of time and resource.

B. Secondly Better solution:

The client initially generates a series of public keys and secret keys. Generate secret for every file upload, compute authenticators and along with file. Drawback: The public key and the secret key are very long and linear. Linear overhead practically unacceptable solution

3.PROPOSED SYSTEM:

In order to overcome form computation overhead and communication overhead should be sub linear to T. To achieve our task we are using binary tree structure to update time periods [28].It guarantees that any authenticator generated in one time period cannot be computed from the secret

International Journal of Research
Available at https://edupediapublications.org/journals
p-ISSN: 2348-6848
e-ISSN: 2348-795X
Volume 03 Issue 17
November 2016

keys for any other time period later than this one. The client can audit the correctness of the cloud data still in aggregated manner, i.e., without retrieve the hole data from the cloud.

System Architecture:

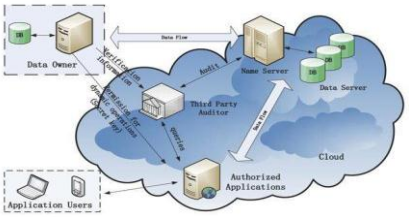

Fig1: system architecture

Figure1 represents the system architecture for our system model. The system having two parties one is data owner and the cloud. The owner produces files and stores these files to the cloud. Cloud is going store these files behalf of owner and provides download service if owner requires. The time of files stored in cloud is T+1 time period, but private key is always unchanged. And secrete key is used for auditing for certain time period.

Algorithms used in Security Model:

1) **AutGen(**PK,j,SK_j,F) (Ø): the authenticator generation algorithm is probabilistic algorithm which takes **SysSetup**(1^k,T) (PK,SK_0): The system setup algorithm is a probabilistic algorithm which takes as input a security parameter k and total number of time period T, and generates private key PK and initial owner's secrete key SK_0. This algorithm runs by the owner.

2) **keyUpdate**(PK,j,SK_j) (SK_{j+1}): The key update algorithm is a probabilistic algorithm which takes as input private key, the current period j and a owner's secret key SK_j, and generates new secret key SK_{j+1}. This algorithm run by the owner.

3) as input the private key PK, the current period j, a owner's secret key SK_j and a file F, and generates the set of authenticators Ø for F in the time period j. this algorithm is run by the owner.

4) **ProofGen**(PK,j,Chal,F,Ø) (P): the proof generation algorithm is a probabilistic algorithm which takes as input private key PK, a time period j, a challenge Chal, a file F and the set of authenticators Ø, and generates a proof P which means the cloud processes F. Here, (j, Chal) pair is issued by the auditor, and then used by the cloud. This algorithm runs by the cloud.

5) **ProofVerify**(PK,j,Chal,P) (True or False): this algorithm is deterministic algorithm which takes as input the private key PK, a time period j, a challenge chal and a proof P, and returns "true" or "false". This algorithm is run by the owner.

IV. RESULTS:

The below tables shows the comparison between existing protocol and the proposed protocol and our performance analysis is proves that protocol is secure and efficiency. Table 1 show the efficiency comparison and table 2 shows complexities of key size and communication overheads. The graph shows the comparison between key generated files, not generated files, users and files.

COMPLEXITIES OF KEYS SIZE AND COMMUNICATION OVERHEADS

Protocols	Secret key size	Public key size	Challenge overhead	Response overhead
Our Protocol	$O((logT)k)$	$O(k)$	$O(k)$	$O((logT)k)$
Shacham *et al.*'s protocol [5]	$O(k)$	$O(k)$	$O(k)$	$O(k)$

Table1

EFFICIENCY COMPARISON

Protocols	SysSetup	KeyUpdate	AuthGen	ProofGen	ProofVerify
Our Protocol	$2T_e$	$4T_e$	$3T_e$	cT_e	$3T_p+(log(T+2)+c+1)T_e$
Shacham *et al.*'s protocol [5]	T_e		$2T_e$	cT_e	$2T_p+cT_e$

Table 2

5.CONCLUSION:

We conclude that our proposed protocol is secure and efficient. The correctness of the data still verified even if owner's current secret key is exposed to cloud storage auditing.

REFERENCES:

[1] Enabling cloud storage auditing with key-Exposure resistance.

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

[3] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Advances in Cryptology-Asiacrypt'08, pp.
90-107, 2008.

[4] C. Erway, A. K¨upc¸ ¨u, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession,". [5] J. Yu, R. Hao, F. Kong, X. Cheng, J Fan, and Y.Chen, "Forward-Secure Identity-Based Signature: Security
Notions and Construction," Information Sciences, Vol. 181, Iss. 3, pp. 648-660, 2011.