www.ioirp.com
International Journal of Innovative Research in Computer Science and Engineering (IJIRCSE)
ISSN: 2394-6364, Volume – 3, Issue – 3. May 2018

Strong Key-Exposure Resilient Auditing for Secure Cloud Storage

V.Anusuya,[1]
Department of CSE,
Greentech college of Engineering for women,
Attur,Salem.
Vnanusha09@gmail.com

V.M.Kirubavathi,[2]
Department of CSE,
Greentech college of Engineering for women,
Attur,Salem.
kiruba17bev@gmail.com

K.Kowsalyaa,[3]
Department of CSE,
Greentech college of Engineering for women,
Attur,Salem.
kowsalyaabe15@gmail.com

S.Parvin Fathima,[4]
Department of CSE,
Greentech college of Engineering for women,
Attur,Salem.
parvinshua@gmail.com

ABSTRACT

The security issue of key exposure is one of the major problems in cloud storage auditing. To overcome this issue, initially the key-exposure resilience scheme had been proposed. However in this scheme, the data from the cloud can be illegally accessed later than the key-exposure time period using the same secret key hat had been provided for auditing the cloud data. An innovative paradigm called strong key-exposure resilient auditing for secure cloud storage which allows to set a particular time period for the key exposure. This preserves the security of the cloud not only earlier but also later than the key exposure time period. The security proof and experimental results demonstrate that our proposed scheme achieves expected security without affecting its efficiency.

Keyword: cloud, cloud computing, data integrity, cloud storage auditing, key exposure.

I. INTRODUCTION

NOWADAYS, cloud storage is the most widely accessed form of choices from individuals to big organizations and enterprises. The cloud computing helps to avoid large storage spaces. Above all it prevents the investment of large capital of users from purchasing and using different hardware's and software's. Although there are tremendous advantages in cloud computing, the security issue of data in the cloud is the significant challenge. The privacy protection of data is an important aspect on shared data[9] of cloud storage auditing. Clients may lose the control of their data and even data loss might happen. Cloud storage auditing is one of the effective security mechanisms [2] to ensure the integrity of data in the cloud. Initially key exposure resisient auditing scheme for secure cloud storage [6] had been proposed. The secret key might be exposed due to low security setting of the client. If the malicious cloud gets the secret key of the client, it can hide the data loss by forging fake data. The malicious clouds can even client's rarely accessed files without being found by the cloud storage auditor.

In order to reduce the computational burden of the client, a third-party auditor (TPA) is introduced to help the client to periodically check the integrity of the data in cloud. Since the key is exposed to the TPA for auditing, the key exposure is another major problem. This key exposure, in some cases, cannot be fully resolved due to the following reasons. When the key-exposure takes place, it cannot be found out at once. The key exposure is always difficult to be found out because the attacker will stop the intrusion at once as he gets the client's secret key. If the attacker does not find the key in a particular time period, he can update the secret key upto the time period in which the key exposure is found. The key-exposure might be detected by the user only when the he finds that the valid authenticators are not generated by himself. At that time, the user has to revoke the old pair of public key and secret key, and generate a new pair.

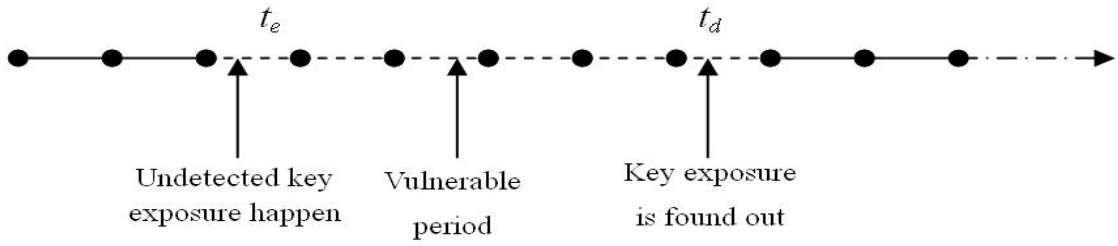

Fig. 1. The security problem between key exposure and its intrusion

II. SYSTEM MODEL

The system comprises three parties: the cloud, the client and the third party auditor (TPA). The cloud offers storage services to the client. The client uploads his files along with the corresponding authenticators to the cloud, and then deletes these data from his storage space. The client can retrieve them from the cloud when he needs them. The TPA is a powerful party and is in charge of two important tasks. The first is to provide auditing service, i.e., periodically check the integrity of the files stored in cloud for the client. The second is to help the client update his secret keys by providing update messages to the client in different time periods. As same as most of public integrity auditing schemes, the TPA is honest for integrity auditing on behalf of cloud users. However, it is not fully trustworthy for key update in our system model. This is because the TPA itself can misuse the key at times. Note that the client's secret information includes two parts. One is the signing secret key SKt, which is used to generate authenticators in time period t. The other is the private key SKc, which is used to generate the signing key SKt in time period t.

The system model is shown in Fig. 2. Assume that one file F stored in cloud is divided into n blocks m_i (i = 1,…., n). The lifetime of file F in our system mode does not need to be fixed initially. It means the total number of time periods in unbounded, which is more close to the reality. The TPA cannot forge the authenticator for any block through his secret key and the update messages he generates. The TPA might incorrectly execute the update key generation algorithm. That is, the TPA might provide the client wrong update key for some reason. In this case, the client should be able to verify the correctness of the update key provided by the TPA. We say a strong key-exposure resilient auditing for secure cloud storage is verifiable if the TPA provides the wrong update key in UMGen algorithm; he cannot pass the verification of our proposed scheme.

III. OUR PROPOSED SCHEME

A. Technical Explanation

In order to achieve the strong key-exposure resilience, we make the signing secret key in each time period be a multiplication of two parts. One part is the update message generated by the TPA, which is computed through the secret key of the TPA and the current time period. The other part is computed from the secret key of the client and the current time period. The signing secret key in any time period must be jointly generated by the client and the TPA. This technique can support both the provable security and the efficient key update. As a result, if the attacker intrudes the client in one time period, he cannot obtain the client's signing secret keys in other time periods without the secret key of the TPA.

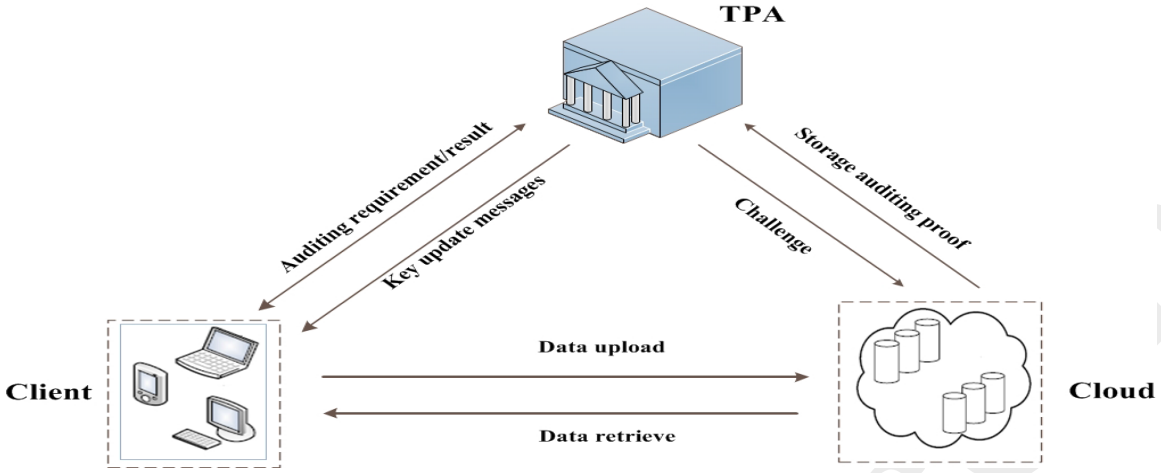

Fig. 2. The proposed system model

B. Algorithms Involved

1. SysSetup (System setup algorithm): This algorithm is run by the client. It takes as input a security parameter k, and generates a system public key PK, the TPA's secret key SK_{TPA} and the client's private key SKc. The client randomly selects SK_C as his private key and PK as his public key.

2. UMGen (Update Message generation algorithm):
This algorithm is run by the TPA at the beginning of each time period. It takes as input the public key PK, the current time period t and the TPA's secret key SKTPA, and generates an update message δ_t. The TPA sends the update message δ_t to the client. The client can verify whether the update message is valid or not.

3. CKeyUpdate (Client key Update algorithm): This algorithm is run by the client at the beginning of each time period. It takes as input the public key PK, the current time period t, the update message δ_t and the client's private key SKc, and generates the signing secret key SKt for time period t.

4. AuthGen (Authenticator generation algorithm): This algorithm is run by the client. It takes as input the public key PK, the current time period t, the client's signing secret key SKt and a file F, and generates a set of authenticators Φ for F in time period t. When the client wants to upload a file F to the cloud in the time period t, he uploads the file tag and the set of authenticators Φ along with the file F to the cloud.

5. ProofGen (Proof Generation algorithm): This algorithm is run by the cloud. It takes as input the public key PK, a time period t, a challenge Chal, a file F and a set of authenticators _, and generates a proof P that is used to prove the cloud stores F correctly. In this algorithm, the (t, Chal) pair is issued by the TPA, and then used by the cloud as input. He TPA first verifies the validity of the file tag. If it is valid, then the TPA selects a subset a an index to be checked. Then the TPA selects random values and sends challenge to the cloud. After receiving the challenge, the cloud replies to the TPA.

6. **ProofVerify** (Proof Verifying algorithm): This algorithm is run by the TPA. It takes as input the public key PK, a time period t, a challenge Chal and a proof P, and returns "true" if the verification passed; or "false", otherwise.

IV. EXPERIMENTAL RESULTS

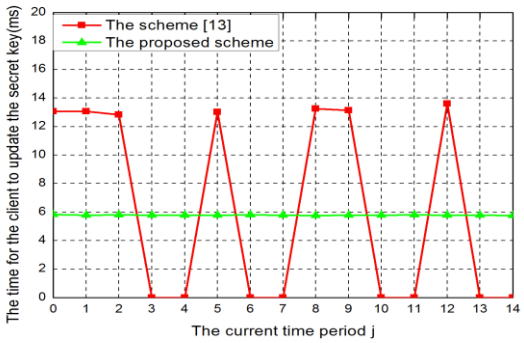

Fig. 3. The key update time in our proposed scheme and the scheme

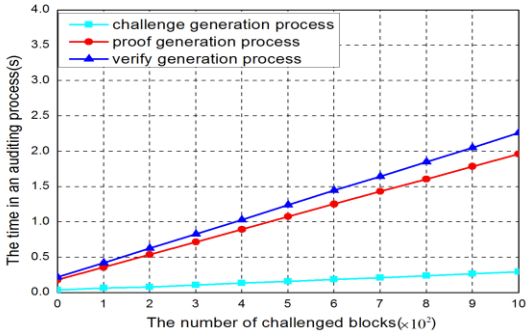

Fig. 4. The time of auditing processes with different number of challenge blocks

The proposed scheme is evaluated in several experiments. These experiments are run on a Linux server with Intel processor running at 2.70 GHz and 4 GB memory. A bilinear map is chosen that uses a supersingular elliptic curve to achieve the fast pairing operations. The base field of this curve is 160 bits, the size of an element in Z_q is 20 bytes, and the size of an element in group G1 is 128 bytes. In our experiments, the data file is set to 20M, which consists of 1,000,000 blocks. In our proposed scheme, the time for the client to update a secret key is independent of the time period because it needs one exponentiation and one multiplication in G1 in any time period. We compare the key update time between the both schemes in Fig.3. the key update time is related to the depth of the node corresponding to the current time period in the full binary tree. When the node is the internal node, the key update time is about 12.6ms; when the node is the leaf node, the key update time is nearly 0ms. In our proposed scheme, the key update time is about 5.8ms in all time periods. It means the client can update the signing secret key using the same computational resource in each time period.

The proposed scheme achieves stronger security without decreasing the efficiency of key updates for the client. In addition, our proposed scheme supports key updates for unlimited time periods. However, the lifetime of the file stored in cloud must be known and fixed for T time periods. When T time periods are over, the cloud storage auditing cannot work any longer. In our experiment, the number of challenged blocks varies from 100 to 1,000. From Fig.4, we can know the challenge generation process spends the least time, which is 0.26s at most; the proof generation process spends a little more time, varying from 0.18s to 1.79s; the proof verification process spends the most time, varying from 0.22s to 2.05s.

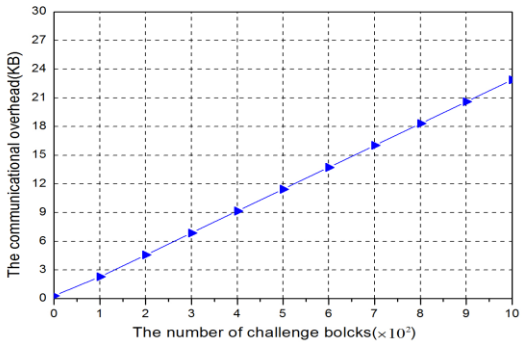

Fig. 5. The challenge overhead with different number number of challenged blocks

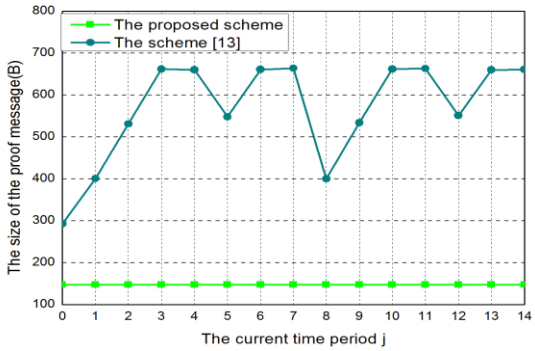

Fig. 6. The proof overhead with different of challenged blocks

In our proposed scheme, the communication overhead consists of the challenge overhead and the proof overhead. The challenge overheads of the proposed scheme and the existing scheme, i.e., cloud storage auditing with key-exposure resistance are Chal= $\{i, v_i\}$, where i is used to determine which blocks will be checked and v_i is used to mix the challenged blocks. Therefore, the proposed scheme and the existing scheme have the same challenge overhead. In Fig.5, when the number of checked blocks varies from 100 to 1,000, the size of the challenge message varies from 2.29KB to 22.89KB. From Fig.6, we can see that the size of the proof message keeps about 148B in all the time periods.

VI.CONCLUSION

This paper has dealt with the key exposure problem in cloud storage auditing using a new paradigm called strong key exposure resilient auditing for cloud storage. This paradigm preserves the security of the cloud not only earlier but also later than the key exposure time period. A definition and the security model of this new kind of cloud storage auditing is formalized. The experimental results demonstrate that the proposed scheme is secure and efficient.

REFERENES

[1] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," IEEE Trans. Parallel and Distributed Systems, Vol. 24, No. 9, pp. 1717-1726, 2013.
[2] K. Yang and X. Jia, "Data Storage Auditing Service in Cloud Computing: Challenges, Methods and opportunities," World Wide Web, vol. 15, no. 4, pp. 409-428, 2012.
[3] Y. Zhu, H.G. Ahn, H. Hu, S.S. Yau, H.J. An, and C.J. Hu, "Dynamic Audit Services for Outsourced Storages in Clouds," IEEE Trans. on Services Computing, vol. 6, no. 2, pp. 409-428, 2013.
[4] Y. Zhu, H. Wang, Z. Hu, G. J. Ahn, H. Hu, and S. S. Yau, "Efficient Provable Data Possession for Hybrid Clouds," Proc. 17th ACM Conference on Computer and Communications Security, pp. 756-758, 2010.
[5] F. Sebe, J. Domingo-Ferrer, A. Martinez-balleste, Y. Deswarte, and J. Quisquater, "Efficient Remote Data Integrity checking in Critical Information Infrastructures," IEEE Transactions on Knowledge and Data Engineering, vol. 20, no. 8, pp. 1-6, 2008.
[6] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling Cloud Storage Auditing with Key-Exposure Resistance,"IEEE Transactions on Information Forensics and Security.vol. 10, no. 6, pp. 1167-1179, Jun. 2015.
[7] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 5, pp. 847-859, May 2011.
[8] C. Wang, S. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," IEEE Trans. Computers, Vol. 62, No. 2, pp. 362- 375, 2013.
[9] B. Wang, B. Li and H. Li. Oruta, "Privacy-Preserving Public Auditing for Shared Data in the Cloud," IEEE Transactions on Cloud Computing, Vol.2, pp. 43-56, 2014.
[10] J. Yuan and S. Yu, "Public Integrity Auditing for Dynamic Data Sharing With Multiuser Modification," IEEE Transactions on Information Forensics and Security, vol. 10, no. 8, pp. 1717-1726, Aug. 2015.