Enabling Cloud Storage Auditing With Key Exposure Resistance

[1]CHINTHA LAVANYA, [2]G.ARUN & [3]V.SRIDHARREDDY
[1]M-Tech, Dept. of CSE Vignana Bharathi Institute Of Technology Ghatkesar, Mail Id: - lavanya.netha18@gmail.com
[2]Associate professor, Dept. of CSE Vignana Bharathi Institute Of Technology Ghatkesar, Mail Id: - garuncse@vbithyd.ac.in
[3]Associate professor, Dept. of CSE Vignana Bharathi Institute Of Technology Ghatkesar, Mail Id: - vsridharreddy19@gmail.com

Abstract

Key-exposure resistance has always been an important issue for in-depth c yber defense in many security applications. Recently, how to deal with the key exposure problem in the settings of cloud storage auditing has been proposed and studied. To address the challenge, existing solutions all require the client to update his secret keys in every time period, which may inevitably bring in new local burdens to the client, especially those with limited computation resources, such as mobile phones. In this paper, we focus on how to make the key updates as transparent as possible for the client and propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this paradigm, key updates can be safely out sourced to some authorized party, and thus the key-update burden on the client will be kept minimal. In particular, we leverage the third party auditor (TPA) in many existing public auditing designs, legit play the role of authorized party in our case, and make it in charge of both the storage auditing and the secure key updates for key-exposure resistance. In our design, TPA only needs to hold an encrypted version of the client's secret key while doing all these burdensome tasks on behalf of the client. The client only needs to download the encrypted secret key from the TPA when uploading new files to cloud. Besides, our design also equips the client with capability to further verify the validity of the encrypted secret keys provided by the TPA. All these salient features are carefully designed to make the whole auditing procedure with key exposure resistance as transparent as possible for the client. We formalize the definition and the security model of this paradigm. The security proof and the performance simulation show that our detailed design instantiations are secure and efficient.

Key words: - Cloud storage, outsourcing computing, cloud storage auditing, key update, verifiability.

1. INTRODUCTION

Cloud computing, as a new technology paradigm with promising further, is becoming more and more popular nowadays. It can provide users with seemingly unlimited computing resource. Enterprises and people can outsource time consuming computation workloads to cloud without spending the extra capital on deploying and maintaining hardware and software. In recent years, outsourcing computation has attracted much attention and been researched widely. It has been considered in many applications including scientific computations, linear algebraic computations, linear programming computations and modular exponentiation computations, etc. Besides, cloud computing can also provide users with seemingly unlimited storage resource. Cloud storage is universally viewed as one of the most important services of cloud computing. Although cloud storage provides great benefit to users, it brings new security challenging problems. One important security problem is how to efficiently check the integrity of the data stored in cloud. In recent years, many auditing protocols for cloud storage have been proposed to deal with this problem. These protocols focus on different aspects of cloud storage uditing such as the high efficiency, the privacy protection of data, the privacy protection of identities, dynamic data operations, the data sharing, etc. The key exposure problem, as another important problem in cloud storage auditing, has been considered recently. The problem itself is non-trivial by nature. Once the client's secret key for storage auditing is exposed to cloud, the cloud is able to easily hide the data loss incidents for maintaining its reputation, even discard the client's data rarely accessed for saving the storage space. Yu et al. constructed a cloud storage auditing protocol with key-exposure resilience by updating the user's secret keys periodically. In this way, the damage of key exposure in cloud storage auditing can be reduced. But it also brings in new local burdens for the client because the client has to execute the key update algorithm in each time period to make his secret key move forward. For some clients with limited computation resources, they might not like doing such extra computations by themselves in each time period. It would be obviously more attractive to make key updates as transparent as possible for the client, especially in frequent key update scenarios. In this paper, we consider achieving this goal by outsourcing key updates. However, it needs to satisfy several new requirements to achieve this goal. Firstly, the real client's secret keys for cloud storage auditing should not be known by the authorized party who performs outsourcing computation for key updates. Otherwise, it will bring the new security threat. So the authorized party should only hold an encrypted version of the user's secret key for cloud storage auditing. Secondly, because the authorized party performing outsourcing computation only knows the encrypted secret keys, key updates should be completed under the encrypted state. In other words, this authorized party should be able to update secret keys for cloud storage auditing from the encrypted version he holds. Thirdly, it should be very efficient for the client to recover the real secret key from the encrypted version that is retrieved from the authorized party. Lastly, the client should be able to verify the validity of the encrypted secret key after the client retrieves it from the authorized party. The goal of this paper is to design a cloud storage auditing protocol that can satisfy above requirements to achieve the outsourcing of key updates. The main contributions are as follows: We propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this new paradigm, key-update operations are not performed by the client, but by an authorized party. The authorized party holds an encrypted secret key of the client for cloud storage auditing and updates it under the encrypted state in each time period. The client downloads the encrypted secret key from the authorized party and decrypts it only when he would like to upload new files to cloud. In addition, the client can verify the validity of the encrypted secret key. We design the first cloud storage auditing protocol with verifiable outsourcing of key updates. In our design, the third party auditor (TPA) plays the role of the authorized party who is in charge of key updates. In addition, similar to traditional public auditing protocols, another important task of the TPA is to check the integrity of the client's files stored in cloud. The TPA does not know the real secret key of the client for cloud storage auditing, but only holds an encrypted version. In the detailed protocol, we use the blinding technique with homomorphism property to form the encryption algorithm to encrypt the secret keys held by the TPA. It makes our protocol secure and the decryption operation efficient. Meanwhile, the TPA can complete key updates under the encrypted state. The client can verify the validity of the encrypted secret key when he retrieves it from the TPA. Therefore, the designed protocol satisfies the above mentioned four requirements. We formalize the definition and the security model of the cloud storage auditing protocol with verifiable outsourcing of key updates. We also prove the security of our protocol in the formalized security model and justify its performance by concrete implementation.

2. RELEGATED WORK

2.1 Existing System

Yu et al. constructed a cloud storage auditing protocol with key-exposure resilience by updating the user's secret keys periodically. In this way, the damage of key exposure in cloud storage auditing can be reduced. But it also brings in new local burdens for the client because the client has to execute the key update algorithm in each time period to make his secret key move forward. For some clients with limited computation resources, they might not like doing such extra computations by themselves in each time period. It would be obviously more attractive to make key updates as transparent as possible for the client, especially in frequent key update scenarios. Wang et al. proposed a public privacy-preserving auditing protocol. They used the random masking technique to make the protocol achieve privacy preserving property.

2.2 Proposed System

We propose a new paradigm called cloud storage auditing with verifiable outsourcing of key updates. In this new paradigm, key-update operations are not performed by the client, but by an authorized party. The authorized party holds an encrypted secret key of the client for cloud storage auditing and updates it under the encrypted state in each time period. The client downloads the encrypted secret key from the authorized party and decrypts it only when he would like to upload new files to cloud. In addition, the client can verify the validity of the encrypted secret key. We design the first cloud storage auditing protocol with verifiable outsourcing of key updates. In our design, the thirdparty auditor (TPA) plays the role of the authorized party who is in charge of key updates. We formalize the definition and the security model of the cloud storage auditing protocol with verifiable outsourcing of key updates. We also prove the security of our protocol in the formalized security model and justify its performance by concrete implementation.

3. IMPLEMENTATION

3.1 SysSetup:

The system setup algorithm is run by theclient. It takes as input a security parameter k and the total number of time periods T , and generates anencrypted initial client's secret key E SK0, a decryptionkey DK and a public key P K . Finally, the client holdsDK , and sends E SK0 to the TPA.

3.2 EkeyUpdate

The encrypted key update algorithm is runby the TPA. It takes as input an encrypted client's secretkey E SK j , the current period j and the public key P K ,and generates a new encrypted secret key E SK j+1 forperiod j + 1.

3.3 Ver ESK:

The encrypted key verifying algorithm is runby the client. It takes as input an encrypted client's secretkey E SK j , the current period j and the public key PK ,if ESK j is a well-formed encrypted client's secret key,returns 1; otherwise, returns 0.

3.4 Dec ESK:

The secret key decryption algorithm is run bythe client. It takes as input an encrypted client's secret key ESK j , a decryption key DK , the current period j and the public key PK , returns the real client's secretkey SK j in this time period.

3.5 AuthGen:

The authenticator generation algorithm is runby the client. It takes as input a file F, a client's secret key SKj , the current period j and the public key P K , and generates the set of authenticators for F in timeperiod j.

3.6 Proof Gen:

The proof generation algorithm is run by thecloud. It takes as input a file F, a set of authenticators ,a challenge Chal, a time period j and the public keyP K , and generates a proof P which proves the cloudstores F correctly.

3.7 Proof Verify:

The proof verifying algorithm is run bythe TPA. It takes as input a proof P, a challenge Chal, a time period j, and the public key P K , and returns**"True" if P is valid; or "False", otherwise.**

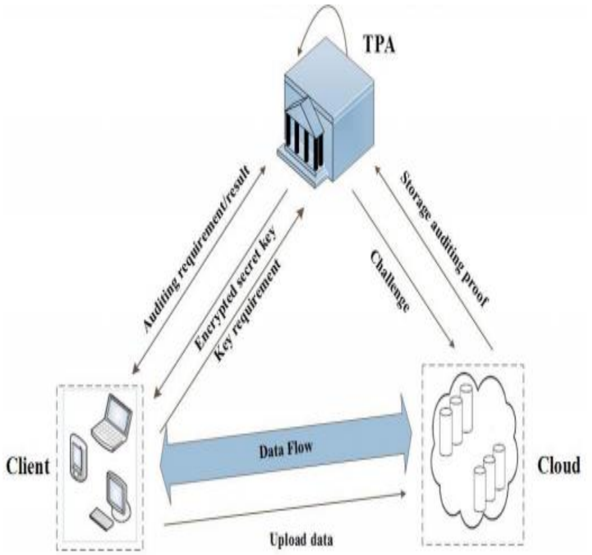
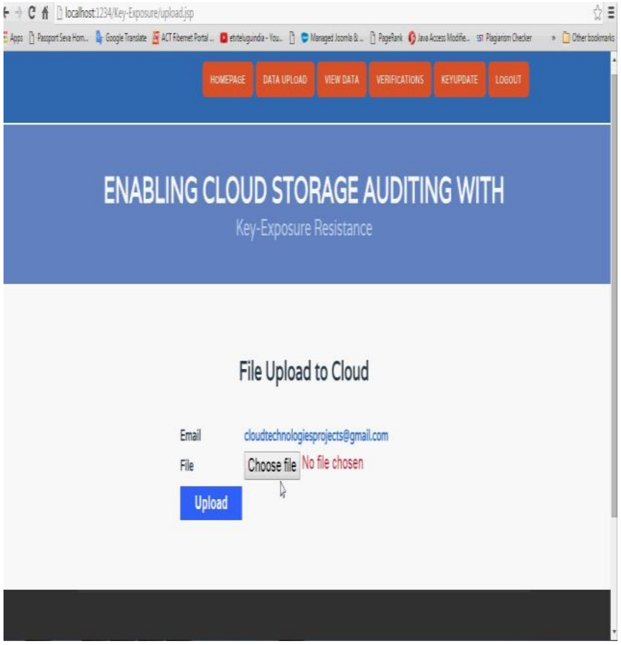


Fig 1 Architecture Diagram

4. EXPERIMENTAL RESULTS



ENABLING CLOUD STORAGE AUDITING WITH

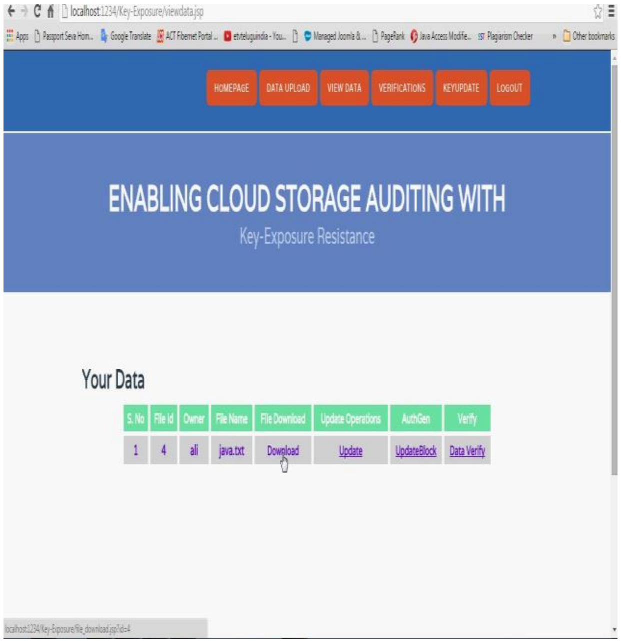
Key-Exposure Resistance

File Upload to Cloud

Email cloudtechnologiesprojects@gmail.com

File No file chosen

Fig 2 File Upload Page


localhost:1234/Key-Exposure/viewdata.jsp
HOMEPAGE
DATA UPLOAD
VIEW DATA
VERIFICATIONS
KEYUPDATE
LOGOUT
localhost:1234/Key-Exposure/file_download.jsp?id=4

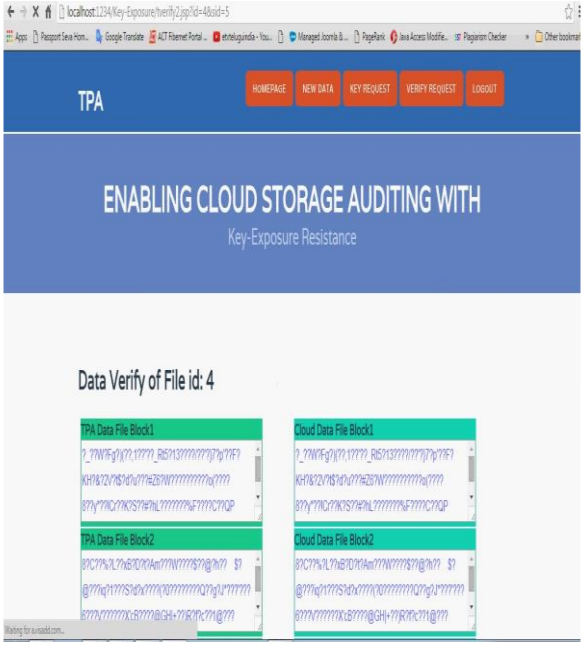
ENABLING CLOUD STORAGE AUDITING WITH

Key-Exposure Resistance

Your Data

S. No	File Id	Owner	File Name	File Download	Update Operations	AuthGen	Verify
1	4	ali	java.txt	Download	Update	UpdateBlock	Data Verify

Fig 3 User View Data Page


localhost:1234/Key-Exposure/verify2.jsp?id=4&sid=5
TPA
HOMEPAGE
NEW DATA
KEY REQUEST
VERIFY REQUEST
LOGOUT
Waiting for avisadd.com...

ENABLING CLOUD STORAGE AUDITING WITH

Key-Exposure Resistance

Data Verify of File id: 4

TPA Data File Block1

```
?_??W?Fg?)(??,1????_Rt5?13????/???)]7?p'??F?
KH?&?2V?t$?d?u???#Z6?W?????????o(????
8??y"??lCr??K?S??#?hL???????%F????C??QP
```

Cloud Data File Block1

```
?_??W?Fg?)(??,1????_Rt5?13????/???)]7?p'??F?
KH?&?2V?t$?d?u???#Z6?W?????????o(????
8??y"??lCr??K?S??#?hL???????%F????C??QP
```

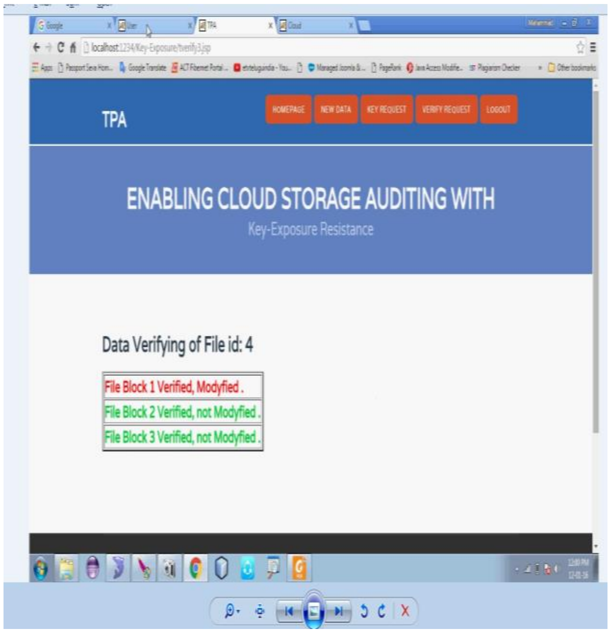
TPA Data File Block2

```
8?C??%?L??xB?D?t?Am???W????$??@?h??   $?
@???iq?1???S?d?x????(?0????????Q??g?J"??????
6???V??????X'cB????@GH|+??|R?f?c??1@???
```

Cloud Data File Block2

```
8?C??%?L??xB?D?t?Am???W????$??@?h??   $?
@???iq?1???S?d?x????(?0????????Q??g?J"??????
6???V??????X'cB????@GH|+??|R?f?c??1@???
```

Fig 4 TPA Data Verify Page

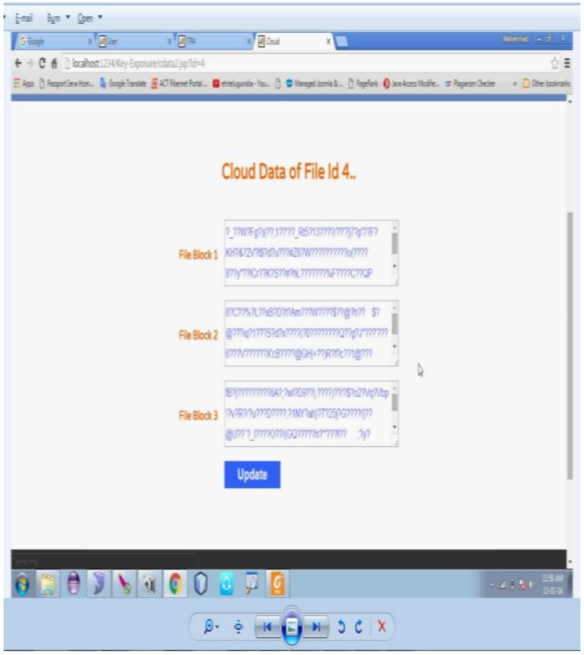


ENABLING CLOUD STORAGE AUDITING WITH
Key-Exposure Resistance

Data Verifying of File id: 4

File Block 1 Verified, Modyfied .
File Block 2 Verified, not Modyfied .
File Block 3 Verified, not Modyfied .

Fig 5 TPA Data Verify Results Page



Cloud Data of File Id 4..

File Block 1

File Block 2

File Block 3

Fig 6 Cloud Data File Updated Page

5. CONCLUSION

In this paper, we study on how to deal with the client's key exposure in cloud storage auditing. We propose a new paradigm called auditing protocol with key-exposure resilience. In such a protocol, the integrity of the data previously stored in cloud can still be verified even if the client's current secret key for cloud storage auditing is exposed. We formalize the definition and the security model of auditing protocol with key-exposure resilience, and then propose the first practical solution. The security proof and the asymptotic performance evaluation show that the proposed protocol is secure and efficient.

6. REFERENCE

[1] 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.

[2] G. Ateniese, R.D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. 4th International Conference on Security and Privacy in Communication Networks, 2008

[3] 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.

[4] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, “MRPPDP: Multiple-Replica Provable Data Possession,” Proc. 28th IEEE International Conference on Distributed Computing Systems, pp. 411-420, 2008.

[5] H. Shacham and B. Waters, “Compact Proofs of Retrievability,” Advances in Cryptology-Asiacrypt’08, pp. 90-107, 2008.

[6] C. Wang, K. Ren, W. Lou, and J. Li, “Toward Publicly Auditable Secure Cloud Data Storage Services,” IEEE Network, vol. 24, no. 4, pp. 19-24, July/Aug. 2010.

[7] 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.

[8] 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.

[9] 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.

[10] 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.