Enabling Cloud Storage Auditing with Key Exposure Resistance

V. Goutham, PhD
HOD Dept. of CSE
TKREC, JNTUH
T.S, India

B. Mounika
Assistant Professor,
Dept. of CSE
TKREC, JNTUH
T.S, India

P. Shiva Datta
M.TECH, Dept. of CSE
TKREC, JNTUH
T.S, India

ABSTRACT
Cloud storage auditing is viewed as an imperative service to corroborate the veracity of the data in public cloud. Existing auditing protocols are all based on the supposition that theClient's secret key for auditing is completely protected. Such assumption may not always be held, due to the probably weak sense of security and/or low security settings at the client. In most of the current auditing protocols would inevitably become unable to work when a secret key for auditing is exposed. It is investigated on how to reduce the damage of the client's key revelation in cloud storage auditing, and provide the first handy elucidation for this new problem setting. Formalized the definition and the security model of auditing protocol with key-exposure resilience and propose such a protocol. Utilized and developed a novel authenticator construction to support the forward security and the property of block less verifiability using the current design. The security proof and the performance analysis show that the projected protocol is protected and well-organized.

Keywords
Data storage, cloud storage auditing, cloud computation, key exposure resistance

1. INTRODUCTION
Cloud computing can help enterprises improve the creation and delivery of IT solutions by providing them with access to services in a cost-effective and flexible manner [2]. Clouds can be classified into three categories, depending on their accessibility restrictions and the deployment model. They are:

- Public Cloud,
- Private Cloud, and
- Hybrid Cloud.

A public Cloud is made available in a pay-as-you-go manner to the general public users irrespective of their origin or affiliation. A private Cloud's usage is restricted to members, employees, and trusted partners of the organization. A hybrid Cloud enables the use of private and public Cloud in a seamless manner. Cloud computing applications span many domains, including business, technology,government, health care, smart grids, intelligent transportation networks, life sciences, disaster management, automation, data analytics, and consumer and social networks. Various models for the creation, deployment, and delivery of these applications as Cloud services have emerged. Cloud storage auditing is used to verify the integrity of the data stored in public cloud, which is one of the important security techniques in cloud storage. In recent years, auditing protocols for cloud storage have attracted much attention and have been researched intensively[16]. These protocols focus on numerous different characteristics of auditing, achieving high bandwidth and computation efficiency is one of the essential concerns. For that perseverance, the Homomorphic Linear Authenticator (HLA) technique that maintains block less verification is explored to diminish the overheads of computation and communication in auditing protocols, which allows the auditor to verify the integrity of the data in cloud without retrieving the whole data. Many cloud storage auditing protocols have been proposed based on this technique. 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.

However, it is possible for the TPA to get the client's data after it executes the auditing protocol multiple times. Auditing protocols in [9] and [10] are designed to ensure the privacy of the client's data in cloud. While all existing protocols focus on the faults or dishonesty of the cloud, they have overlooked the possible weak sense of security and/or low security settings at the client. The procedure to deal with the client's secret key exposure for cloud storage auditing is a very important problem. It is focused here on how to reduce the damage of the clients key exposure in cloud storage auditing.

The process involves the downloading of whole data from the cloud, producing new authenticators, and re-uploading everything back to the cloud, all of which can be tedious and cumbersome in designing a cloud storage auditing protocol with built-in key-exposure resilience. Besides, it cannot always guarantee that the cloud provides real data when the client regenerates new authenticators. Unswervingly espousing Standard key-evolving technique is also not suitable for the new problem setting. It can lead to repossessing all of the actual files blocks when the verification is proceeded. This is partly because the technique is incompatible with block less verification. The resulting authenticators cannot be accrued, leading to unacceptably high computation and communication cost for the storage auditing [6].

2. RELATED WORK
In order to check the integrity of the data stored in the remote server, many protocols were proposed [14] These protocols focused on various requirements such as high efficiency, stateless verification, data dynamic operation, privacy protection, etc. According to the role of the auditor, these auditing protocols can be divided into two categories: private verification and public verification. In an auditing protocol with private verifiability, the auditor is provided with a secret that is not known to the proven or other parties. Only the auditor can verify the integrity of the data. In contrast, the verification algorithm does not need a secret key from the auditor in an auditing protocol with public verifiability.

Therefore, any third party can play the role of the auditor in this kind of auditing protocols. Atenieseet al. [1] firstly considered the public verification and proposed the notion of "provable data possession" (PDP) for ensuring data possession at untrusted storages. They used the technique of HLA and random sample to audit outsourced data. Juels and Kaliski Jr. explored a "proof of retrievability" (PoR) model. They used the tools of spot-checking and error-correcting codes to ensure both possession and retrievability of files on remote storage systems. Shacham and Waters [5] gave two short and efficient homomorphic authenticators: one has private verifiability which is based on pseudorandom functions; the other has public verifiability which is based on the BLS signature. Dodiset al. [31] focused on the study on different variants of existing POR work. Shah et al. introduced a TPA to keep online storage honest. The protocol requires the auditor to maintain the state, and suffers from bounded usage. Wang et al. [10] provided a public auditing protocol that has privacy-preserving property. In order to make the protocol achieve privacy-preserving property, they integrate the HLA with random masking technique. Wang proposed a proxy provable data possession protocol. In this protocol, the client delegates its data integrity checking task to a proxy. Dynamic data operations for audit services are also attended in order to make auditing more flexible. Atenieseet al. [2] firstly proposed a partially dynamic PDP protocol. Wang et al. [11] proposed another auditing protocol supporting data dynamics.

In this protocol, they utilized the BLS-based HLA and Merkle Hash Tree to support fully data dynamics. Erwayet al. [13] extended the PDP model and proposed a skip list-based protocol with dynamics support. Zhu et al. proposed a cooperative provable data possession protocol which can be extended to support the dynamic auditing. Yang and Jia [9] proposed a dynamic auditing protocol with privacy-preserving property. The problem of user revocation in cloud storage auditing was considered in [15]. Most of above auditing protocols are all built on the assumption that the secret key of the client is absolutely secure and would not be exposed. But as we have shown previously, this assumption may not always be true. The current work advances the field by exploring how to achieve key-exposure resistance in cloud storage auditing, under the new problem settings.

3. SYSTEM DESIGN

The first study has been done on how to achieve the key-exposure resilience in the storage auditing protocol and propose a new concept called auditing protocol with key-exposure resilience [4]. In such a protocol, any dishonest behaviors, such as deleting or modifying some client's data stored in cloud in previous time periods, can all be detected, even if the cloud gets the client's current secret key for cloud storage auditing [9]. This very important issue is not addressed before by previous auditing protocol designs.

We further formalize the definition and the security model of auditing protocol with key-exposure resilience for secure cloud storage. We design and realize the first practical auditing protocol with built-in key-exposure resilience for cloud storage. In order to achieve current goal, we employ the binary tree structure, seen in a few previous works [4] on different cryptographic designs, to update the secret keys of the client. Such a binary tree structure can be considered as a variant of the tree structure used in the HIBE scheme [9]. In addition, the pre-order traversal technique is used to associate each node of a binary tree with each time period. In current detailed protocol, the stack structure is used to realize the pre-order traversal of the binary tree. We also design a novel authenticator supporting the forward security and the property of block less verifiability.

We prove the security of current protocol in the formalized security model, and justify its performance via concrete asymptotic analysis. Indeed, the proposed protocol only adds reasonable overhead to achieve the key-exposure resilience. We also show that current proposed design can be extended to support the TPA, lazy update and multiple sectors. An auditing system for secure cloud storagein Fig. 1. The system involves two parties: the client (files owner) and the cloud. The client produces files and uploadsthese files along with corresponding authenticators to thecloud. The cloud stores these files for the client and provides download service if the client requires. Each file is furthermoredivided into multiple blocks [2]. For the simplicity of description,The client can periodically audit whether his files in cloud arecorrect. The lifetime of files stored in the cloud is divided intoT + 1 time periods . In currentmodel, the client will update his secret keys for cloud storageauditing in the end of each time period, but the public keyis always unchanged. The cloud is allowed to get the client'ssecret key for cloud storage auditing in one certain time period.It means the secret key exposure can happen in this systemmodel.An auditing protocol with key-exposureresilience is composed by five algorithms (SysSetup,Key Update, AuthGen, ProofGen, ProofVerify), Current security model considers the notion of the forwardsecurity [11] and data possession property [1]. In Table I, we indicate a game to describe an adversary A against thesecurity of an auditing protocol with key-exposure resilience.

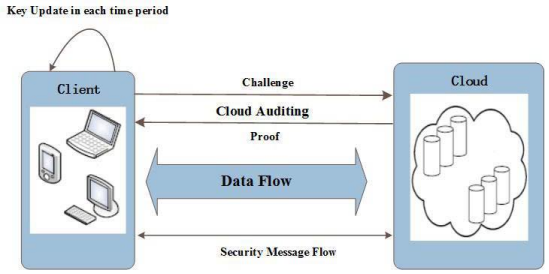

Fig. 1.1. System model of current cloud storage auditing

An auditing system for secure cloud storagein Fig. 1. The system involves two parties: the client (files owner) and the cloud. The client produces files and uploads these files along with corresponding authenticators to the cloud. The cloud stores these files for the client and provides download service if the client requires. Each file is furthermoredivided into multiple blocks [2]. For the simplicity of description, The client can periodically audit whether his files in cloud are correct. The lifetime of files stored in the cloud is divided into T + 1 time periods . In current model, the client will update his secret keys for cloud storage auditing in the end of each time period, but

the public keyis always unchanged. The cloud is allowed to get the client's secret key for cloud storage auditing in one certain time period. It means the secret key exposure can happen in this systemmodel. An auditing protocol with key-exposureresilience is composed by five algorithms (SysSetup,KeyUpdate, AuthGen, ProofGen, ProofVerify), Current security model considers the notion of the forward security [11] and data possession property [1]. In Table I, we indicate a game to describe an adversary A against the security of an auditing protocol with key-exposure resilience.

Table 1: A game to describe an adversaryagainstthe security of the protocol

Game Adversary-Forge(A)

$j = 0; A \xleftarrow{PK} SysSetup(\cdot);$

Repeat

$A \leftarrow AuthGen_{SK_j}(F = (m_1, ..., m_n));$

$SK_{j+1} \leftarrow KeyUpdate(SK_j); j \leftarrow j+1;$

Until A goes to the break-in phase;

$b = j; A \leftarrow SK_b;$

$A \leftarrow (Chal, j^*(j^* < b));$

$P = A(Forge, F, Chal, j^*(j^* < b));$

if ProofVerify$(PK, j^*, Chal, P) = 1$

then return "*True*" *else* return "*False*".

The above security model captures that an adversary cannot forge a valid proof for a time period prior to keyexposure without owning all the blocks corresponding to a given challenge, if it cannot guess all the missing blocks.In each time period prior to key exposure, the adversary isallowed to query the authenticators of all the blocks. Theadversary can be given a secret key for auditing in thekey-exposure (break-in) time period.

4. PROPOSED SYSTEM

We firstly show two basic solutions for the key-exposure problem of cloud storage auditing before we give current core protocol. The first is a ingenuous solution, which in fact cannot fundamentally solve this problem. In this solution, the client still uses the traditional key revocation method. Once the client knows his secret key for cloud storage auditing 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[8]. 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[7].

The second is a slightly better solution, which can solve this problem but has a large overhead. They are both impractical when applied in realistic settings. And then we give current core protocol that is much more efficient than both of the basic solutions. Current goal is to design a practical auditing protocol withkey-exposure resilience, in which the operational complexitiesof key size, computation overhead and communication overheadshould be at most sub linear to T. In order to achieve currentgoal, we use a binary tree structure to appoint time periods and associate periods with tree nodes by the pre-order traversal technique [14]. The secret key in each time period is organizedas a stack. In each time period, the secret key is updatedby a forward-secure technique [18]. It guarantees that anyauthenticator generated in one time period cannot be computedfrom the secret keys for any other time period later than thisone. Besides, it helps to ensure that the complexities of keyssize, computation overhead and communication overhead areonly logarithmic in total number of time periods T.

As aresult, the auditing protocol achieves key-exposure resiliencewhile satisfying current efficiency requirements. As it will be shownlater, in current protocol, the client can audit the integrity of thecloud data still in aggregated manner, i.e., without retrievingthe entire data from the cloud. As same as the key-evolvingmechanisms [11]–[13], current protocol does not considerthe key exposure resistance during one time period.

The public key in current protocol is denoted by PK which isfixed during the whole lifetime. In current protocol, each node ofthe binary tree corresponding to j has one key pair (S**wj**,R**wj**),where S**wj**is called as the node secret key which is used to generate authenticators and R**wj**is called as verification valuewhich is used to verify the validity of authenticators. The keypair of the root node is denoted by (S, R). The client's secretkey SK j in period j is composed by two parts X j and _j .The first part X j is a set composed by the key pair (S**wj**,R**wj**)and the key pairs of the right siblings of the nodes on thepath from the root to **wj**. That is, if w_0 is a prefix of **wj**,then X j contains the secret key (Sw_1, Rw_1). In current protocol,the first part X j is organized as a stack satisfying first-in firstoutprinciple with (S**wj**,R**wj**) on top. The stack is initiallyset (S, R) in time period 0. The second part _j is composedby the verification values from the root to node **wj**exceptthe root. So Ω_j= (R**wj**|1, . . . ,R**wj**|t) when **wj**= w1 · · ·wt.

Description of Current Protocol:

1) SysSetup: Input a security parameter k and the total time period T. Then

a) Run IG(1k) to generate two multiplicative groups G1, G2 of some prime order q and an admissible pairingˆe : G1 × G1 → G2.

b) Choose cryptographic hash functions H1 :G1 → G1, H2 : {0, 1}∗ × G1 → Z∗ qand H3 : {0, 1}∗ × G1 → G1. Select two independent generators g, u ∈G1.

c) The client selects ρ ∈Z∗qat random, and computes R = gρand S = H1(R)ρ.

Fig 1.2. An example to show the stack changes from time period 0 to time period 9 when l = 4.

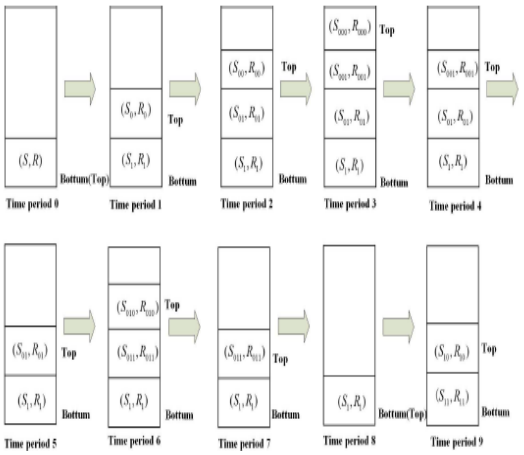

Table 1.2: Efficiency comparison

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	$2T_p + cT_e$

Current proposed protocol can easily be modified to supportthe TPA because we have considered the public verificationduring current design. In the modified auditing protocol supportingthe TPA, the SysSetupalgorithm, the Key Update algorithmand the AuthGenalgorithm are the same as the descriptionin Section 3. In the Proof Gen algorithm, we only modify currentoriginal protocol as follows: The TPA generates a challengeChal= {(i, vi)}i∈I, and sends it to the cloud. After the cloudcompletes the same operations as those in original protocolin Section 3, it sends the proof P to the TPA instead of theclient. In the Proof Veri f y algorithm, we only need to makethe TPA instead of the client verify the validity of the tag andthe proof P[19].The block less verifiability means that the cloud canconstruct a proof that allows the auditor to check the integrityof certain file blocks in cloud, even when the auditor does nothave access to the actual file blocks.

5. CONCLUSION

In the proposed paradigm, it is deliberated on how to deal with the client's key exposure in cloud storage auditing. A new standard called auditing protocol with key-exposure resilience. The integrity of the data formerly stored in cloud can still be substantiated even if the client's current secret key for cloud storage auditing is bare in these kinds of protocols. It is enacted in thedefinition and the security model of auditing protocol with key-exposure resilience, and has given the practical solution. The security proof and the asymptotic presentation assessment depicted that the protocol is secure and efficient. The efficient comparison betweencurrent protocol and earlier protocol based on BLS signature also has been provided.

6. REFERENCES

[1] G. Atenieseet al., "Provable data possession at untrusted stores,"inProc. 14th ACM Conf. Comput. Commun.Secur., 2007,pp. 598–609.

[2] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable andefficient provable data possession," in Proc. 4th Int. Conf. Secur. Privacy Commun.Netw., 2008, Art. ID 9.

[3] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, andJ.-J. Quisquater, "Efficient remote data possession checking in criticalinformation infrastructures," IEEE Trans. Knowl. Data Eng., vol. 20,no. 8, pp. 1034–1038, Aug. 2008.

[4] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiplereplicaprovable data possession," in Proc. 28th IEEE Int. Conf. Distrib.Comput. Syst., Jun. 2008, pp. 411–420.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability,"in Advances in Cryptology—ASIACRYPT. Berlin, Germany:Springer-Verlag, 2008, pp. 90–107.

[6] C. Wang, K. Ren, W. Lou, and J. Li, "Toward publicly auditable securecloud data storage services," IEEE Netw., vol. 24, no. 4, pp. 19–24,Jul./Aug. 2010.

[7] Y. Zhu, H. Wang, Z. Hu, G.-J.Ahn, H. Hu, and S. S. Yau, "Efficientprovable data possession for hybrid clouds," in Proc. 17th ACM Conf.Comput.Commun.Secur., 2010, pp. 756–758.

[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 protocolfor data storage in cloud computing," IEEE Trans. Parallel Distrib. Syst.,vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[10] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacypreservingpublic auditing for secure cloud storage," IEEE Trans.Comput., vol. 62, no. 2, pp. 362–375, Feb. 2013.

[11] B. G. Kang, J. H. Park, and S. G. Hahn, "A new forward securesignature scheme," Cryptology ePrint Archive, Tech. Rep. 2004/183,2004. [Online]. Available: http://eprint.iacr.org/2004/183

[12] J. Yu, F. Kong, X. Cheng, R. Hao, and G. Li, "One forward-securesignature scheme using bilinear maps and its applications," Inf. Sci.,vol. 279, pp. 60–76, Sep. 2014.

[13] J. Yu, R. Hao, F. Kong, X. Cheng, J. Fan, and Y. Chen, "Forwardsecureidentity-based signature: Security notions and construction," Inf.Sci., vol. 181, no. 3, pp. 648–660, 2011.

[14] C. Gentry and A. Silverberg, "Hierarchical ID-based cryptography,"in Advances in Cryptology—ASIACRYPT. Berlin, Germany:Springer-Verlag, 2002, pp. 548–566.

[15] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability forlarge files," in Proc. 14th ACM Conf. Comput. Commun.Secur., 2007,pp. 584–597.

[16] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability viahardness amplification," in Proc. 6th Theory Cryptogr. Conf., 2009,pp. 109–127.

[17] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing tokeep online storage services honest," in Proc. 11th USENIX WorkshopHot Topics Oper. Syst., 2007, pp. 1–6.

[18] Y. Zhu, H. Hu, G.-J.Ahn, and M. Yu, "Cooperative provable datapossession for integrity verification in multicloud storage," IEEETrans. Parallel Distrib. Syst., vol. 23, no. 12, pp. 2231–2244,Dec. 2012.

7. AUTHOR PROFILE

Dr V. Goutham is a Professor and Head of the Department of computer Science and Engineering at TKR Engineering College affiliated to J.N.T.U Hyderabad. He received M.Tech from Andhra University and B.Tech from J.N.T.U Hyderabad. He worked for various MNC Companies in Software Testing and Quality as Senior Test Engineer. His research interests are Software Reliability Engineering, software testing, software Metrics, and cloud computing.

Ms. B.Mounika is a Assistant Professor in the Department of computer Science and Engineering at TKR Engineering College affiliated to J.N.T.U Hyderabad.

Mr.P.SHIVA DATTA Department of computer Science and Engineering at TKR Engineering College affiliated to J.N.T.U Hyderabad.