Analysis of One Scheme for Enabling Cloud Storage Auditing with Verifiable Outsourcing of Key Updates

Zhengjun Cao[1], Lihua Liu[2], Olivier Markowitch[3]
(Corresponding author: Zhengjun Cao)

Department of Mathematics, Shanghai University[1]
No.99, Shangda Road, Shanghai, China
(Email: caozhj@shu.edu.cn)
Department of Mathematics, Shanghai Maritime University[2]
No.1550, Haigang Ave, Pudong New District, Shanghai, China
Computer Sciences Department, Université Libre de Bruxelles[3]
Boulevard du Triomphe CP 212, 1050 Bruxelles, Belgique
(Received July 25, 2016; revised and accepted Nov. 15 & Dec. 25, 2016)

Abstract

Cloud computing supports a paradigm shift from local to network-centric computing and enables customers with limited computational resources to outsource large-scale computational tasks to the cloud, such as linear equations and linear programming. Recently, Yu et al. [IEEE TIFS, 11(6), 2016, 1362-1375] have proposed a scheme for cloud storage auditing with verifiable outsourcing of key updates. In this note, we remark that Yu et al.'s scheme has two inherent weaknesses: 1) it does not truly mitigate the client's computational burden for key updates; 2) it does not ensure confidentiality since the files uploaded to the cloud by the client are eventually not encrypted at all.

Keywords: Cloud Computing; Cloud Storage Auditing; Confidentiality; Third-party Auditor

1 Introduction

Cloud computing making use of the tremendous resources of computing and storage systems via the Internet, supports a paradigm shift from local to network-centric computing [17, 19], and benefits scientific and engineering applications, such as data mining and many other computational and data-intensive activities. It enables customers with limited computational resources to outsource large-scale computational tasks to the cloud, including linear equations [21, 23], linear programming [9, 20, 22], bilinear pairing [1, 7, 8, 11], matrix inversion computation [14] and matrix multiplication computation [13]. Many researchers have studied the new computing paradigm and proposed a lot of schemes [2, 6, 10, 12, 15, 16]. But some kinds of flaws in some outsourcing schemes [3, 4, 5] were found for security, efficiency or other reasons.

Key-exposure problem is a special one related to key management. A primary observation on the problem in the scenario of cloud storage auditing is that once the client's secret key for storage auditing is exposed to the cloud, the cloud is able to easily hide the data loss incidents for maintaining its reputation. More seriously, the cloud maybe discard the client's data rarely accessed for saving the storage space [24].

Very recently, Yu et al. [25] have proposed a scheme for cloud storage auditing with verifiable outsourcing of key updates. The scheme involves three entities, the client, the cloud infrastructure and a third-party auditor (TPA). The client uploads his files to the cloud. The TPA audits the integrity of the files stored in the cloud and regularly updates the encrypted secret keys of the client to prevent any secret key exposition. Upon receiving the returned encrypted secret key, the client decrypts it to get the real secret key.

In this note we would like to stress that in Yu et al.'s scheme the recovered secret key is only used to generate authenticators for the files instead of protecting these files. No secret key associated to a symmetric key encryption is dedicated to the protection of the files. Besides, the scheme does not truly mitigate the client's computational burden for key updates. In view of these weaknesses, we would like to point out that Yu et al.'s scheme could not be practically implemented.

2 Review of Yu et al.'s Scheme

The scheme [25] uses the following notations. G_1, G_2 are two multiplicative groups with some prime order q. g is a generator of G_1. $\hat{e} : G_1 \times G_1 \to G_2$ is a bilinear pairing. $H_1 : G_1 \to G_1, H_2 : \{0,1\}^* \times G_1 \to \mathbb{Z}_q^*$ and $H_3 : \{0,1\}^* \times G_1 \to G_1$ are three cryptographic hash functions. T is the total periods number of the whole lifetime for the files stored in the cloud. $w_1 \cdots w_t$ is the binary string of the node w^j associated with period j. $w^j|_k (k \le t)$ is the k-prefix of w^j. w^{j0}, w^{j1} are the left child node and the right child node of w^j, respectively. $w^j|_{\bar{k}}$ is the sibling node of $w^j|_k$. PK is the public key which is unchanged in the whole lifetime. ES_{w^j} is the encrypted node secret key. R_{w^j} is the verification value which is used to verify the validity of authenticators. ESK_j is the client's encrypted secret key in period j. X_j is the set composed by the key pairs. Ω_j is a set composed by the verification values. (ES, R) is the key pair of the root node. F is a file which the client wants to store in cloud. $m_i (i = 1, \cdots, n)$ are n blocks of file F. DK is the decryption key to recover the encrypted secret key for cloud storage auditing.

It involves three entities, some clients, a third-party auditor (TPA), and the cloud. The client has a secret key associated to a signature $SSig$ for ensuring the integrity of not only the file identifier $name$ but also the time period j. The protocol consists of the following phases:

SysSetup: Given a security parameter k and the total time period T, the client picks a generator u of G_1, $\rho, \tau \in \mathbb{Z}_q^*$ and computes $R = g^\rho, G = g^\tau, ES = H_1(R)^{\rho - \tau}$. Set $PK = (R, G, u)$. Set $X_0 = \{(ES, R)\}, \Omega_0 = \emptyset$ (where $\emptyset$ is null set), $DK = \tau$ and keep it himself. The client sends $ESK_0 = (X_0, \Omega_0)$ to the TPA.

EkeyUpdate: Input ESK_j, the time period j, and the public key PK. The TPA parses $ESK_j = (X_i, \Omega_j)$ where X_j is organized as a stack which consists of (ES_{w^j}, R_{w^j}) and the key pairs of the right siblings of the nodes on the path from the root to w^j. The top element of the stack is (ES_{w^j}, R_{w^j}). Firstly, pop (ES_{w^j}, R_{w^j}) off the stack. Then do as follows:

- If w^j is an internal node ($w^{j+1} = w^{j0}$ in this case), select $\rho_{w^{j0}}, \rho_{w^{j1}} \in \mathbb{Z}_q^*$. And then compute $R_{w^{j0}} = g^{\rho_{w^{j0}}}, R_{w^{j1}} = g^{\rho_{w^{j1}}}$, $h_{w^{j0}} = H_2(w^{j0}, R_{w^{j0}}), h_{w^{j1}} = H_2(w^{j1}, R_{w^{j1}})$, $ES_{w^{j0}} = ES_{w^j} \cdot H_1(R)^{\rho_{w^{j0}} h_{w^{j0}}}$, $ES_{w^{j1}} = ES_{w^j} \cdot H_1(R)^{\rho_{w^{j1}} h_{w^{j1}}}$. Push $(ES_{w^{j1}}, R_{w^{j1}})$ and $(ES_{w^{j0}}, R_{w^{j0}})$ onto the stack orderly. Let X_{j+1} denote the current stack and define $\Omega_{j+1} = \Omega_j \bigcup \{R_{w^{j0}}\}$.
- If w^j is a leaf, define X_{j+1} with the current stack. If $w_t = 0$ (the node w_{j+1} is the right sibling node of w^j in this case), then set $\Omega_{j+1} = \Omega_j \bigcup \{R_{w^{j+1}}\} - \{R_{w^j}\}$ ($R_{w^{j+1}}$ can be read from the new top $(ES_{w^{j+1}}, R_{w^{j+1}})$ of the stack). If $w_t = 1$ ($w^{j+1} = w''1$ in this case, where w'' is the longest string such that $w''0$ is a prefix of w^j), then set $\Omega_{j+1} = \Omega_j \bigcup \{R_{w^{j+1}}\} - \{R_{w''0}, R_{w''01}, \cdots, R_{w_t}\}$ ($R_{w^{j+1}}$ can be read from the new top $(ES_{w^{j+1}}, R_{w^{j+1}})$ of the stack).
- Erase key pair (ES_{w^j}, R_{w^j}), and return $ESK_{j+1} = (X_{j+1}, \Omega_{j+1})$.

VerESK: Input a client's encrypted secret key $ESK_j = (X_j, \Omega_j)$, the current period j and the public key PK. The client checks that

$$\hat{e}(g, ES_{w^j}) = \hat{e}\left(R/G \cdot \prod_{m=1}^{t} R_{w^{j1}}^{h_{w^{j1}}}, H_1(R)\right),$$

where $h_{w^j} = H_2(w^j, R_{w^j})$.

DecESK: The client computes $S_{w^j} = ES_{w^j} \cdot H_1(R)^\tau$. The real secret key is set as $SK_j = (X'_j, \Omega_j)$, where X'_j is the same stack as X_j except that the top element in X'_j is (S_{w^j}, R_{w^j}) instead of (ES_{w^j}, R_{w^j}) in X_j.

AuthGen: For a file $F = \{m_1, \cdots, m_n\}$ and the current period j, the client proceeds as follows.

- Parse $SK_j = (X'_j, \Omega_j)$ and read the top element (S_{w^j}, R_{w^j}) from the stack X'_j. Select $r \in \mathbb{Z}_q^*$ and compute $U = g^r$,

$$\sigma_i = H_3(name\|i\|j, U)^r \cdot S_{w^j} \cdot u^{rm_i}$$

  ($i = 1, \cdots, n$), where the $name$ is chosen randomly from $\mathbb{Z}_q^*$ as the identifier of the file F. Generate a file tag for F and j using the signature $SSig$ in order to ensure the integrity of $name$ and j. Denote the set of authenticators in time period j with $\Phi = (j, U, \{\sigma_i\}_{1 \le i \le n}, \Omega_j)$.
- Send the file F and the set of authenticators along with the file tag to cloud.

ProofGen: Input a file F, a set of authenticators $\Phi = (j, U, \{\sigma_i\}_{1 \le i \le n}, \Omega_j)$, a time period j, a challenge $Chal = \{(i, v_i)\}_{i \in I}$ (where $I = \{s_1, \cdots, s_c\}$ is a c-element subset of set $[1, n]$ and $v_i \in \mathbb{Z}_q$) and the public key PK. The cloud calculates an aggregated authenticator $\Phi = (j, U, \sigma, \Omega_j)$, where $\sigma = \Pi_{i \in I} \sigma_i^{v_i}$. It also computes $\mu = \sum_{i \in I} v_i m_i$. It then sends $P = (j, U, \sigma, \mu, \Omega_j)$ along with the file tag as the response proof of storage correctness to the TPA.

ProofVerify: Input a proof P, a challenge $Chal$, a time period j and the public key PK. The TPA parses $\Omega_j = (R_{w^j|_1}, \cdots, R_{w^j|_t})$. He then verifies the integrity of $name$ and j by checking the file tag. After that, the client verifies whether the following equation holds:

$$\begin{aligned} \hat{e}(g, \sigma) &= \hat{e}(R \cdot \Pi_{m=1}^{t} R_{w^j|_m}^{h_{w^j|_m}}, H_1(R)^{\sum_{i \in I} v_i}) \\ &\quad \cdot \hat{e}(U, u^\mu \cdot \Pi_{i \in I} H_3(name\|i\|j, U)^{v_i}), \end{aligned}$$

Table 1: Yu et al.'s outsourcing scheme

Client		TPA		Cloud
–*SysSetup*. Set $PK = (R, G, u)$, $X_0 = \{(ES, R)\}, \Omega_0 = \emptyset$. $DK = \tau, ESK_0 = (X_0, \Omega_0)$.	$\xrightarrow{ESK_0}$ $\xleftarrow{ESK_1}$ $\vdots$	–EkeyUpdate. Update it as $ESK_1 = (X_1, \Omega_1)$.		
–*VerESK*. Check $\hat{e}(g, ES_{\omega^j}) \overset{?}{=} \hat{e}(R/G \cdot \prod_{m=1}^{t} R_{\omega_{j_1}}{}^{h_{\omega^{j_1}}}, H_1(R))$.	$\xleftarrow{ESK_j}$			
–*DecESK*. Compute $S_{\omega^j} = ES_{\omega^j} \cdot H_1(R)^{\tau}$, $SK_j = (X'_j, \Omega_j)$.				
–*AuthGen*. For a file $F = \{m_1, \cdots, m_n\}$, compute $U = g^r$, $\sigma_i = H_3(name\|i\|j, U)^r \cdot S_{\omega^j} \cdot u^{rm_i}$ $i = 1, \cdots, n$. Generate *tag* for F, j using the signature *SSig*.				
Set $\Phi = (j, U, \{\sigma_i\}_{1 \le i \le n}, \Omega_j)$.	$- - -$	upload F, Φ, *tag* $- - - - - - - - - - - - - -$	$\rightarrow$	Store F, Φ, *tag*.
		–*Challenge*. Set $Chal = \{(i, v_i)\}_{i \in I}$.	$\xrightarrow{Chal}$	–*ProofGen*. $\sigma = \prod_{i \in I} \sigma_i^{v_i}$, $\mu = \sum_{i \in I} v_i m_i$.
		–*ProofVerify*. Verify the integrity of *name* and j by checking the *tag*. Check $\hat{e}(g, \sigma) \overset{?}{=} \hat{e}\left(R \cdot \prod_{m=1}^{t} R_{\omega^j\|m}^{h_{\omega^j\|m}}, H_1(R)^{\sum_{i \in I} v_i}\right) \cdot \hat{e}\left(U, u^{\mu} \cdot \prod_{i \in I} H_3(name\|i\|j, U)^{v_i}\right)$	$\xleftarrow{P}$	$P = \{j, U, \sigma, \mu, \Omega_j\}$.

where $h_{w^j} = H_2(w^j, R_{w^j})$. If it holds, returns "True", otherwise returns "False".

We refer to the following Table 1 for a brief description of Yu et al.'s scheme [25].

3 Analysis of Yu et al.'s Scheme

The scheme [25] aims to deal with the key exposure problem. They proposed the paradigm of cloud storage auditing which enables a client to outsource the burden of key updates to the third party auditor. But we find the scheme has two inherent flaws.

1) *The scheme does not truly mitigate the client's computational burden for key updates.* Concretely, in the time period j, the client's main computational task is to calculate

$$\begin{aligned} U &= g^r,\ S_{\omega^j} = ES_{\omega^j} \cdot H_1(R)^{\tau}, \\ \sigma_i &= H_3(name\|i\|j, U)^r \cdot S_{\omega^j} \cdot u^{rm_i}, \\ &\qquad i = 1, \cdots, n \\ \hat{e}(g, ES_{\omega^j}) &\overset{?}{=} \hat{e}(R/G \cdot \prod_{m=1}^{t} R_{\omega_{j_1}}{}^{h_{\omega^{j_1}}}, H_1(R)). \end{aligned}$$

while the TPA's main computational task is to calculate

$$R_{\omega^{j_0}} = g^{\rho_{\omega^{j_0}}},\ ES_{\omega^{j_0}} = ES_{\omega^j} \cdot H_1(R)^{\rho_{\omega^{j_0}} h_{\omega^{j_0}}},$$

$$R_{\omega^{j_1}} = g^{\rho_{\omega^{j_1}}},\ ES_{\omega^{j_1}} = ES_{\omega^j} \cdot H_1(R)^{\rho_{\omega^{j_1}} h_{\omega^{j_1}}},$$

$$\begin{aligned}\hat{e}(g,\sigma) &\overset{?}{=} \hat{e}\left(R\cdot\prod_{m=1}^{t} R_{\omega^{j}|m}^{h_{\omega^{j}|m}}, H_1(R)^{\sum_{i\in I} v_i}\right)\\ &\cdot\hat{e}\left(U, u^{\mu}\cdot\prod_{i\in I} H_3(name\|i\|j,U)^{v_i}\right).\end{aligned}$$

It is easy to find that the computational task for the client is almost equal to that for the TPA. That means the client's computational burden is not truly alleviated.

2) *The scheme does not ensure any confidentiality since the files uploaded by the client to the cloud are in fact not encrypted at all.* In the phase *AuthGen*, it is specified that the client has to parse the file F as $\{m_1,\cdots,m_n\}$ and generate the authenticator Φ by invoking these m_i. The client then sends $\{F,\ \Phi,\ tag\}$ to the cloud. Clearly, the authors have forgotten to assign a symmetric key encryption to protect the file F. In practice, it is conventional to encrypt files using any symmetric key encryption, such as AES. Due to computing overhead [18], a public key encryption is usually just used to establish the secure channel needed to exchange the secret key of a symmetric-key system.

3) We would like to stress that if the uploaded file F is viewed as an encrypted file, then the scheme has to assign three groups of secret keys, SK_1 for encrypting the file F, SK_2 *for generating the authenticator* Φ *for* F, and SK_3 for signing $name$ and the timestamp j (it specifies that "the client has held a secret key for a signature $SSig$, which is used to ensure the integrity not only of the file identifier $name$ but also of the time period j"). Obviously, both SK_1 and SK_3 held by the client could also be exposed (as SK_2), but the authors [25] have not realized the main danger. The proposed method only solves the problem of updating SK_2. Therefore, *the proposed scheme does not solve correctly the client's key exposure problem.*

4) We think it seems impossible to revise the scheme such that it could simultaneously update the involved secret keys SK_1, SK_2 and SK_3, because the client has to retrieve the file F stored previously on the cloud in order to update the symmetric key SK_1. In such case, it is totally unnecessary to introduce the key SK_2 for generating the authenticator Φ for the file F. Frankly speaking, this is an inherent flaw in the proposed model by Yu et al.

4 Conclusion

We show that there are two flaws in Yu et al.'s scheme for cloud storage auditing with verifiable outsourcing of key updates, and remark that the scheme cannot be practically implemented. We would like to stress that the proposed paradigm is somewhat artificial because the client is able to update his secret keys solely by himself.

Acknowledgments

The work is supported by the National Natural Science Foundation of China (61303200, 61411146001), the Brussels Region (Innoviris) and the SeCloud Project. We are grateful to the reviewers for their valuable suggestions.

References

[1] S. Canard, J. Devigne, and O. Sanders, "Delegating a pairing can be both secure and efficient," in *Proceedings of Applied Cryptography and Network Security (ACNS'14)*, pp. 549–565, Lausanne, Switzerland, June 2014.

[2] S. Canard et al., "Toward generic method for server-aided cryptography," in *Proceedings of Information and Communications Security (ICICS'13)*, pp. 373–392, Beijing, China, November 2013.

[3] Z. J. Cao and L. H. Liu, "Comment on 'harnessing the cloud for securely outsourcing large-scale systems of linear equations'," *IEEE Transactions on Parallel Distributed Systems*, vol. 27, no. 5, pp. 1551–1552, 2016.

[4] Z. J. Cao and L. H. Liu, "A note on two schemes for secure outsourcing of linear programming," *International Journal of Network Security*, vol. 19, no. 2, pp. 323–326, 2017.

[5] Z. J. Cao, L. H. Liu, and O. Markowitch, "On two kinds of flaws in some server-aided verification schemes," *International Journal of Network Security*, vol. 18, no. 6, pp. 1054–1059, 2016.

[6] F. Chen, T. Xiang, and Y. Y. Yang, "Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud," *Journal of Parallel Distributed Computing*, vol. 74, pp. 2141–2151, 2014.

[7] X. F. Chen et al., "Efficient algorithms for secure outsourcing of bilinear pairings," *Theoretical Computer Science*, no. 562, pp. 112–121, 2015.

[8] B. Chevallier-Mames et al., "Secure delegation of elliptic-curve pairing," in *Proceedings of Smart Card Research and Advanced Application, 9th IFIP WG 8.8/11.2 International Conference (CARDIS'10)*, pp. 24–35, Passau, Germany, Apr. 2010.

[9] J. Dreier and F. Kerschbaum, "Practical privacy-preserving multiparty linear programming based on problem transformation," in *Proceedings of IEEE International Conference on Privacy, Security, Risk, and Trust, and IEEE International Conference on Social Computing (PASSAT/SocialCom'11)*, pp. 916–924, Boston, MA, USA, Oct. 2011.

[10] M. Girault and D. Lefranc, "Server-aided verification: Theory and practice," in *Proceedings of Advances in Cryptology (ASIACRYPT'05)*, pp. 605–623, Chennai, India, Dec. 2005.

[11] S. Hohenberger and A. Lysyanskaya, "How to securely outsource cryptographic computations," in

Proceedings of Theory of Cryptography (TCC'05), pp. 264–282, Cambridge, MA, USA, Feb. 2005.

[12] W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for secure data storage in cloud computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133–142, 2016.

[13] X. Y. Lei, X. F. Liao, T. W. Huang, and F. Heriniaina, "Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud," *Information Sciences*, vol. 280, pp. 205–217, 2014.

[14] X. Y. Lei, X. F. Liao, T. W. Huang, H. Q. Li, and C. Q. Hu, "Outsourcing large matrix inversion computation to a public cloud," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 78–87, 2013.

[15] C. W. Liu, W. F. Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing," *International Journal of Network Security*, vol. 18, no. 4, pp. 650–666, 2016.

[16] J. Liu, C. K. Chu, and J. Y. Zhou, "Identity-based server-aided decryption," in *Proceedings of Information Security and Privacy (ACISP'11)*, pp. 337–352, Melbourne, Australia, July 2011.

[17] D. Marinescu, *Cloud Computing Theory and Practice*. USA: Elsevier, 2013.

[18] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*, New York, U.S.A.: CRC,Taylor & Francis, 1996.

[19] A. Mosa, H. M. El-Bakry, S. M. Abd El-Razek, S. Q. Hasan, "A proposed E-government framework based on cloud service architecture," *International Journal of Electronics and Information Engineering*, vol. 5, no. 2, pp. 93–104, 2016.

[20] H. X. Nie, X. F. Chen, J. Li, J. Liu, and W. J. Lou, "Efficient and verifiable algorithm for secure outsourcing of large-scale linear programming," in *Proceedings of 28th IEEE International Conference on Advanced Information Networking and Applications (AINA'14)*, pp. 591–596, Victoria, BC, Canada, May 2014.

[21] S. Salinas, C. Q. Luo, X. H. Chen, and P. Li, "Efficient secure outsourcing of large-scale linear systems of equations," in *Proceedings of 2015 IEEE Conference on Computer Communications (INFOCOM'15)*, pp. 1035–1043, Hong Kong, China, Apr. 2015.

[22] C. Wang, K. Ren, and J. Wang, "Secure optimization computation outsourcing in cloud computing: A case study of linear programming," *IEEE Transactions on Computers*, vol. 65, no. 1, pp. 216–229, 2016.

[23] C. Wang, K. Ren, J. Wang, and Q. Wang, "Harnessing the cloud for securely outsourcing large-scale systems of linear equations," *IEEE Transactions on Parallel Distributed Systems*, vol. 24, no. 6, pp. 1172–1181, 2013.

[24] Z. Wang, Y. Lu, G. Sun, "A policy-based deduplication mechanism for securing cloud storage," *International Journal of Electronics and Information Engineering*, vol. 2, no. 2, pp. 70–79, 2015.

[25] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics Security*, vol. 11, no. 6, pp. 1362–1375, 2016.

Biography

Zhengjun Cao is an associate professor with the Department of Mathematics at Shanghai University. He received his Ph.D. degree in applied mathematics from Academy of Mathematics and Systems Science, Chinese Academy of Sciences. His research interests include applied cryptography, discrete algorithms and quantum computation.

Lihua Liu is an associate professor with the Department of Mathematics at Shanghai Maritime University. She received her Ph.D. degree in applied mathematics from Shanghai Jiao Tong University. Her research interests include combinatorics and cryptography.

Olivier Markowitch is an associate professor with the Computer Sciences Department at the Universite Libre de Bruxelles. He is also information security advisor of his University. He is working on the design and analysis of two-party and multi-party cryptographic protocols as well as on the design and analysis of digital signature schemes.