On the Security of a Public Auditing Mechanism for Shared Cloud Data Service

Yong Yu, Jianbing Ni, Man Ho Au, Yi Mu, Boyang Wang, and Hui Li

***Abstract*—Recently, a public auditing protocol for shared data called Panda (IEEE Transactions on Services Computing, doi: 10.1109/TSC.2013.2295611) was proposed to ensure the correctness of the outsourced data. A distinctive feature of Panda is the support of data sharing and user revocation. Unfortunately, in this letter, we show that Panda is insecure in the sense that a cloud server can hide data loss without being detected. Specifically, we show that even some stored file blocks have been lost, the server is able to generate a valid proof by replacing a pair of lost data block and its signature with another block and signature pair. We also provide a solution to the problem while preserving all the desirable features of the original protocol.**

Keywords: Cloud storage, data integrity, security analysis.

I. Introduction

Data storage and sharing services, such as Dropbox and Google Drive, enable data owners to conveniently access, modify and share data in a user group. Although this promising approach offers great convenience to cloud users, it does trigger many security threats towards users' data [1], one of which being data loss and leakage. As noted in [2], the data hosted on cloud might be compromised due to hardware/software failures and human errors. It is commonly assumed that the cloud server is semi-trusted. On the one hand, it is willing to provide data storage services for its users. On the other hand, it is not trusted to report data loss incidents as it could damage its reputation [3].

To protect the integrity of data in the cloud, several novel auditing mechanisms [3], [4] have been presented. These mechanisms allow the data owner or an auditor to verify that the data are properly stored. They assume the owner is a single entity and is thus not applicable to the situation where data sharing is needed. One issue that prevent these mechanisms to be used for data sharing is the necessity of user revocation in case data are to be shared. Recently, a public auditing mechanism named Panda [5] was proposed to support data sharing among users and multitask auditing. This mechanism, derived from homomorphic authenticable proxy re-signature, can significantly improve the efficiency of user revocation.

In this letter, we investigate the public auditing mechanism in [5] and demonstrate that the construction fails to achieve its goal since it suffers from the *replace attack*. To be more specific, if a data block m_t and its signature σ_t is discarded, the cloud server can use another valid pair of data block and its signature $(m_{t'}, \sigma_{t'})$ to replace (m_t, σ_t) to generate an auditing proof. Consequently, the server can hide data loss accidents by fooling the verifier to believe that the shared data are well maintained as long as a small fraction of the data is still available. We then give a simple but adequate approach to remedy the weakness without losing any features of the original protocol.

Yong Yu and Jianbing Ni are with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, 611731, China. Email: yyucd2012@gmail.com.

Yong Yu and Yi Mu are with School of Computer Science and Software Engineering, University of Wollongong, Wollongong, NSW 2522, Australia.

Man Ho Au is with Department of Computing, The Hong Kong Polytechnic University, Hong Kong.

Boyang Wang and Hui Li are with the State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, Shaanxi, 710071, China.

II. Review of Panda

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups with the same prime order p. g and ω are two generators of $\mathbb{G}_1$. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denotes a bilinear map and $H : \{0,1\}^* \to \mathbb{G}_1$, $H' : \{0,1\}^* \to Z_p^*$ represent two cryptographic hash functions. The global parameters are $(\mathbb{G}_1, \mathbb{G}_2, p, \hat{e}, g, \omega, H, H')$. The data $M = (m_1, \cdots, m_n)$ are shared among d users $(u_1, \cdots, u_d)$, where n is the total number of blocks. Here we briefly review the protocol in [5].

KeyGen. User u_i selects a random $\pi_i \in \mathbb{Z}_p^*$ as his private key sk_i and computes the public key $pk_i = g^{\pi_i}$. Assume u_1 is the original user, who is the creator of the shared data.

ReKey. The cloud server generates a re-signing key $rk_{i\to j} = \pi_j/\pi_i$ following the approach suggested by Ateniese et al. [6].

Sign. Given the private key π_i, a data block $m_k \in \mathbb{Z}_p (k \in [1,n])$ and its block identifier $id_k = \{v_k || H'(m_k||v_k)||s_k\}$, where v_k is the virtual index of the block and s_k is the signer identifier of m_k, u_i outputs a signature on m_k as:

$$\sigma_k = (H(id_k)\omega^{m_k})^{\pi_i} \in \mathbb{G}_1.$$

ReSign. Given $rk_{i\to j}$, pk_i, σ_k, m_k and id_k, the server checks $\hat{e}(\sigma_k, g) \overset{?}{=} \hat{e}(H(id_k)\omega^{m_k}, pk_i)$. If not, the server outputs $\perp$; Otherwise, outputs

$$\sigma'_k = \sigma_k^{rk_{i\to j}} = (H(id_k)\omega^{m_k})^{\pi_i \cdot \pi_j/\pi_i} = (H(id_k)\omega^{m_k})^{\pi_j}.$$

ChalGen. A verifier generates a challenge as follows:

1) Randomly pick a c-element subset L of set $[1,n]$.
2) Select a random $\eta_l \in \mathbb{Z}_q^*$ for $l \in L$ and q is a smaller prime than p.
3) Send the auditing challenge $\{l, \eta_l\}_{l\in L}$ to the server.

ProofGen. Upon receiving $\{l, \eta_l\}_{l\in L}$, the server generates a proof of possession for the shared data M. Specifically,

1) The server divides the set L into d subsets $(L_1, \cdots, L_d)$ based on the signer identifier of each selected block, where L_i is the subset of selected blocks signed by user u_i. The number of elements in subset L_i is c_i. Clearly,

we have $c = \sum_{i=1}^{d} c_i$, $L = L_1 \cup \cdots \cup L_d$ and $L_i \cap L_j = \emptyset$ for $i \neq j$.

2) For each subset L_i, the server computes

$$\alpha_i = \textstyle\sum_{l \in L_i} \eta_l m_l \in \mathbb{Z}_q, \quad \beta_i = \prod_{l \in L_i} \sigma_l^{\eta_l} \in \mathbb{G}_1.$$

3) The server responds with an auditing proof $\{\alpha, \beta, \{id_l, s_l\}_{l \in L}\}$ to the verifier, where $\alpha = (\alpha_1, \cdots, \alpha_d)$ and $\beta = (\beta_1, \cdots, \beta_d)$.

ProofVerify. Given the auditing challenge $\{l, \eta_l\}_{l \in L}$, the proof $\{\alpha, \beta, \{id_l, s_l\}_{l \in L}\}$ and the public keys of all the existing users $(pk_1, \cdots, pk_d)$, the verifier checks

$$\hat{e}(\textstyle\prod_{i=1}^{d} \beta_i, g) \overset{?}{=} \prod_{i=1}^{d} \hat{e}(\prod_{l \in L_i} H(id_l)^{\eta_l} \cdot \omega^{\alpha_i}, pk_i).$$

If it holds, the verifier outputs 1; otherwise, it outputs 0.

III. On the Security of Panda

Some comprehensive security analyses are provided in [5]. However, we found that Panda suffers from the replace attack and thus fails to achieve the claimed security goals. A replace attack enables an adversary to choose a valid and uncorrupted pair of block and its signature to replace a challenged but corrupted or deleted pair. Consequently, the server is able to fool the verifier into believing the shared data are well maintained while in reality a majority of data blocks have been lost. Consider a file of 10000 blocks. Our result implies that the server can cheat the verifier with a probability of 99% (the same threshold of 99% is used in [2] as a requirement for a correct protocol) as long as the number of corrupted blocks is smaller than 9540.

Suppose a challenged block m_t of the file M, which belongs to u_θ, has been polluted or deleted, but an unchallenged block $m_{t'}$ and its signature $\sigma_{t'}$ signed by $u_{\theta'}$ are well maintained on the cloud, where $\theta, \theta' \in [1, d]$. The server will perform a replace attack and generate a response as follows:

1) Upon receiving the auditing message $\{l, \eta_l\}_{l \in L}$ from the verifier, where $\{t, \eta_t\} \in \{l, \eta_l\}_{l \in L}$ and $\{t', \eta_{t'}\} \notin \{l, \eta_l\}_{l \in L}$, the server divides L into d subsets $L = (L_1, \cdots, L_d)$ and the number of elements in L_i is c_i.
2) For each subset L_i, the server performs as follows.
   - if $i = \theta$, compute
   $$\alpha_i = \textstyle\sum_{l \in L_i \setminus t} \eta_l m_l, \quad \beta_i = \prod_{l \in L_i \setminus t} \sigma_l^{\eta_l}.$$
   - if $i = \theta'$, compute
   $$\alpha_i = \textstyle\sum_{l \in L_i} \eta_l m_l + \eta_t m_{t'}, \quad \beta_i = \prod_{l \in L_i} \sigma_l^{\eta_l} \cdot \sigma_{t'}^{\eta_t}.$$
   - if $i \neq \theta$ and $i \neq \theta'$, compute
   $$\alpha_i = \textstyle\sum_{l \in L_i} \eta_l m_l, \quad \beta_i = \prod_{l \in L_i} \sigma_l^{\eta_l}.$$
3) The server outputs an auditing proof
$$\{\alpha, \beta, \{id_l, s_l\}_{l \in L \setminus t}, \{id_{t'}, s_{\theta'}\}\}$$
to the verifier, where $\alpha = (\alpha_1, \cdots, \alpha_d)$ and $\beta = (\beta_1, \cdots, \beta_d)$.

The proof can make the verification equation hold since

$$\begin{aligned}\hat{e}(\prod_{i=1}^{d} \beta_i, g) &= \hat{e}(\prod_{l \in L_i, i=\theta} H(id_l)^{\eta_l} \cdot \omega^{\alpha_i}, pk_i) \cdot \\ &\quad \hat{e}(\prod_{l \in L_i, i=\theta'} H(id_l)^{\eta_l} \cdot \omega^{\alpha_i}, pk_i) \cdot \\ &\quad \prod_{i=1, i\neq\theta,\theta'}^{d} \hat{e}(\prod_{l \in L_i} H(id_k)^{\eta_l} \cdot \omega^{\alpha_i}, pk_i) \\ &= \prod_{i=1}^{d} \hat{e}(\prod_{l \in L_i} H(id_l)^{\eta_l} \cdot \omega^{\alpha_i}, pk_i).\end{aligned}$$

The attack works because all the elements in the verification are provided by the cloud server. As a result, the server can always generate a response that makes the equation holds by utilizing the uncorrupted blocks and the corresponding signatures. Although the virtual indices v_k are involved in block identifiers id_k to keep the right order for each block, it does not resist the replace attack.

To overcome this weakness, we suggest using a kind of authenticated data structure, such as rank-based authenticated skip list or Merkle Hash Tree (MHT), to authenticate the signatures such that the signatures protect the integrity of file blocks while the authenticated data structure ensures the integrity and security of signatures. This approach can prevent a cheating server from substituting an unspotted pair of the block and the signature for a spotted one. The successful example of exploiting MHT to enforce block sequence and achieve data dynamics simultaneously can be found in [3]. The fixed protocol preserves all the properties of the original protocol such as efficient user revocation, public auditing and scalability. In terms of the performance, the extra computation overhead that comes from the hash function used in generating the MHT is light and the server only needs to store MHT structure additionally. Regarding the communication cost, the user has to upload the root of MHT to the cloud and the server needs to respond the values of the challenged nodes and their siblings required to calculate the root of MHT in the auditing process. The computation expenditure and the communication cost are acceptable for both the cloud users and the server according to the experimental results in [3].

IV. Conclusion

In this letter, we revisited a public auditing mechanism for shared data with efficient user revocation [5] and demonstrated that by utilizing the replace attack, the server can forge a response to a challenge to cheat the verifier that the shared data in cloud are properly stored, while the data have been corrupted. We also suggested a solution to remedy this weakness without losing any features of the original protocol.

References

[1] Cloud Security Alliance, "Top Threats to Cloud Computing," http://www.cloudsecurityalliance.org, 2010.

[2] G. Ateniese, R. C. Burns, R. Curtmola, J. Herring, L. Kissner, Z. N. J. Peterson, and D. X. Song, "Provable Data Possession at Untrusted Stores," in *Processing of ACM Conference on Computer and Communications Security (CCS2007)*, 2007, pp. 598–609.

[3] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li,"Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[4] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and S. Chen, "Dynamic Audit Services for Outsourced Storage in Clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227–238, 2013.

[5] B. Wang, B. Li, and H. Li, "Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud," *IEEE Transactions on Services Computing*, doi: 10.1109/TSC.2013.2295611, 2013.

[6] G. Ateniese, and S. Hohenberger, "Proxy Re-Signatures: New Definitions, Algorithms, and Applications," in: *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS 2005)*, 2005, pp. 310–319.