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 in the cloud server can hide data loss without being detected. Specifically, we are able to generate a valid proof by replacing a pair of lost data block and its signature to replace (m_i, \sigma_i) 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.

II. REVIEW OF PANDA

Let \(G_1\) and \(G_2\) be two multiplicative cyclic groups with the same prime order \(p\). \(g\) and \(\omega\) are two generators of \(G_1\), \(\hat{e} : G_1 \times G_2 \rightarrow G_2\) denotes a bilinear map and \(H : \{0, 1\}^* \rightarrow G_1, H' : \{0, 1\}^* \rightarrow Z_p^*\) represent two cryptographic hash functions. The global parameters are \((G_1, G_2, p, \hat{e}, g, \omega, H, H')\). The data \(M = (m_1, \ldots, m_n)\) are shared among \(d\) users \((u_1, \ldots, u_d)\), where \(n\) is the total number of blocks. Here we briefly review the protocol in [5].

KeyGen. User \(u_i\) selects a random \(\pi_i \in Z_p^*\) 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 \(r_{k_{i \rightarrow j}} = \pi_j / \pi_i\) following the approach suggested by Ateniese et al. [6].

Sign. Given the private key \(\pi_i\), a data block \(m_k \in Z_p\) and its block identifier \(id_k = \{v_k|H'(mk)|v_k\}\), where \(v_k\) is the virtual index of the block and \(sk_k\) is the signer identifier of \(m_k\), \(u_i\) outputs a signature on \(m_k\) as:

\[
\sigma_k = (H(id_k)\omega^{mk})^{\pi_i} \in G_1.
\]

ReSign. Given \(r_{k_{i \rightarrow j}}, pk_i, \sigma_k, m_k\) and \(id_k\), the server checks \(\hat{e}(\sigma_k, g) = \hat{e}(H(id_k)\omega^{mk}, pk_i)\). If not, the server outputs \(\bot\); Otherwise, outputs \(\sigma_k' = \sigma_k^{r_{k_{i \rightarrow j}}} = (H(id_k)\omega^{mk})^{\pi_i / \pi_j} = (H(id_k)\omega^{mk})^{\pi_j}\).

ChalGen. A verifier generates a challenge as follows:
1) Randomly pick a \(c\)-element subset \(L\) of set \([1, n]\).
2) Select a random \(\eta_l \in Z_p^*\) 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 \((L_1, \ldots, L_d)\) 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, its goal since it suffers from the replace attack. To be more specific, if a data block \(m_i\) and its signature \(\sigma_i\) is discarded, the cloud server can use another valid pair of data block and its signature \((m', \sigma_i')\) to replace \((m_i, \sigma_i)\) to generate an auditing proof.
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

$$a_i = \sum_{t \in L_i} m_t \epsilon Z_{q_t}, \quad \beta_i = \prod_{t \in L_i} \sigma_{t}^{m_t} \in G_1.$$

3) The server responds with an auditing proof \(\{\alpha, \beta, \{id_t, s_t\}_{t \in L}\}\) to the verifier, where

\[\alpha = (\alpha_1, \ldots, \alpha_d)\quad \text{and}\quad \beta = (\beta_1, \ldots, \beta_d).\]

**ProofVerify.** Given the auditing challenge \(\{l, \eta_l\}_{l \in L}\), the proof \(\{\alpha, \beta, \{id_t, s_t\}_{t \in L}\}\) and the public keys of all the existing users \((pk_1, \ldots, pk_d)\), the verifier checks

$$e(\prod_{i=1}^{d} \beta_i, g) = \prod_{i=1}^{d} \hat{e}(\prod_{t \in L_i} H(id_t)^{m_t} \cdot \omega^{s_t}, pk_i).$$

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 $u_{\theta}$, 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 $\theta, \theta' \in \{1, \ldots, d\}$. 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 \(\{t, \eta_t\}_{t \in L} \neq \{l, \eta_l\}_{l \in L}\) 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
  $$\alpha_i = \sum_{t \in L_i \setminus \{t\}} m_t, \quad \beta_i = \prod_{t \in L_i \setminus \{t\}} \sigma_{t}^{m_t}.$$

- if $i = \theta'$, compute
  $$\alpha_i = \sum_{t \in L_i} m_t + m_{t'}, \quad \beta_i = \prod_{t \in L_i} \sigma_{t}^{m_t} \cdot \sigma_{t'}^{m_{t'}}.$$

- if $i \neq \theta$ and $i \neq \theta'$, compute
  $$\alpha_i = \sum_{t \in L_i} m_t, \quad \beta_i = \prod_{t \in L_i} \sigma_{t}^{m_t}.$$

3) The server outputs an auditing proof \(\{\alpha, \beta, \{id_t, s_t\}_{t \in L_i \setminus \{t\}}, \{id_{t'}, s_{t'}\}\}\) 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

$$\hat{e}(\prod_{i=1}^{d} \beta_i, g) = \hat{e}(\prod_{t \in L_i, i=\theta} H(id_t)^{m_t} \cdot \omega^{s_t}, pk_i).$$

$$\hat{e}(\prod_{t \in L_i, i=\theta'} H(id_t)^{m_t} \cdot \omega^{s_t}, pk_i).$$

$$\hat{e}(\prod_{i=1}^{d} \beta_i, g) = \hat{e}(\prod_{i=1}^{d} \beta_i, g) = \prod_{i=1}^{d} \hat{e}(\prod_{t \in L_i} H(id_t)^{m_t} \cdot \omega^{s_t}, pk_i).$$

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 $q_t$ are involved in block identifiers $id_t$ 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 [6]. 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 [6].

## 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


