On the Security of a Public Auditing Protocol for Shared Data with Efficient User Revocation in the Cloud

FangChao Ma[1] ,Hechao Li[2],Hua Guo[2,a],ChunHe Xia[1]
1. Beijing Key Laboratory of Network Technology, Beihang University, Beijing 100191, China
2. State Key Laboratory of Software Development Environment,Beihang University, Beijing 100191, China
3. [a] Corresponding Author: hguo.xyz@163.com

Abstract.

Recently, Wang et. al. proposed a public auditing protocol with efficient user's revocation for shared data in the cloud storage (InfoCom 2013, 2904-2912). By taking a careful look at the scheme, we noticed that their scheme has inherent limitations. In short, In short, the auditor cannot find the correct public key to verify those re-signatures which are converted from the original signatures. In this paper, we show a detailed weakness analysis on Wang et al.'s protocol, and propose a solution to remedy the weakness without sacrificing any desirable features of the mechanism.

Keywords: Public auditing, Shared data, Cloud storage, User revocation, Cryptanalysis

Introduction

Recently, sharing data with each other in a group is very popular with the development of data storage and sharing services provided by the cloud. In this work model, an original group user creates shared data and hosts them on the cloud server. After that, every group user is able to access and modify shared data so that he can share the latest version with the rest of the group. To protect shared data's integrity, all of the data, including the data created by the original group user and the data modified by different group users, should be signed before they

are stored on the cloud server. Thus different data blocks are signed by different users due to data modifications performed by different users. Since each data block is signed by a group user, a public verifier, such as a third party auditor (TPA), can check data integrity in the cloud without downloading the entire data, referred to as public auditing.

When a user is revoked from the group, he is no longer able to access and modify shared data, and the signatures generated by him are no longer valid to the group [1]. Therefore, although the content of shared data is not changed during user revocation, the blocks signed by the revoked user have to be re-signed by an existing user in the group, so that, after the revocation, the integrity of the entire data can still be verified with the public keys of existing users only.

Most of the previous works [2, 3, 4, 5] focus on auditing the integrity of personal data and preserve identity privacy from the TPA. None of them considers the efficiency of user revocation when auditing the correctness of shared data until recently, Wang et al. [6] presented a public auditing mechanism with efficient user revocation in an untrusted cloud by utilizing proxy re-signatures. More precisely, once a user is revoked, the cloud is able to re-sign the blocks, which were signed by the revoked user, with a re-signing key. Meanwhile, the cloud, which is not in the same trusted domain with each user, is only able to convert the signature of the revoked user into a signature of an existing user on the same block, but it cannot sign arbitrary blocks on behalf of either the revoked user or an existing user.

In this paper, we revisit Wang et al.'s [6] public auditing protocol and show that there is an inherent limitation on their protocol. In short, the auditor cannot find the correct public key to verify those re-signatures which are converted from the original signatures. we show a detailed weakness analysis on Wang et al.'s protocol, and propose a solution to remedy the weakness without sacrificing any desirable features of the mechanism.

Review the Protocol

Let G_1 and G_2 be two groups of order *p*, *g* be a generator of G_1, $e: G_1 \times G_1 \rightarrow G_2$ be a bilinear map, w be a random element of G_1. The global parameters are $(e, p, G_1, G_2, g, w, H, H')$, where H is a hash function with $H: \{0,1\}^* \rightarrow G_1$ and H' is a hash function with $H': \{0,1\}^* \rightarrow Z_q$. The total number of blocks in shared data is n, and shared data is described as $M = (m_1, ..., m_n)$. The total number of users in the group is d. Without loss of generality, assume user u_1 is the original user, who is the creator of shared data. Wang et.al's public auditing mechanism consists of six algorithms: KeyGen, ReKey, Sign, ReSign, ProofGen, ProofVerify.

- **KeyGen.** User u_i generates a random $x_i \in Z_p$, and outputs his/her public key $pk_i = g^{x_i}$ and private key $sk_i = x_i$. The original user also creates a user list (UL), which contains ids of all the users in the group. The user list is public and signed by the original user.
- **ReKey.** Assume that private and authenticated channels exist between each pair of entities, and there is no collusion.To generate a re-signing key k_{ij}, the cloud firstly selects a random $r \in Z_p$ and sends it to user u_i. After that, User u_i sends r / x_i to user u_j, where $sk_i = x_i$. Then user u_j sends rx_j / x_i to the cloud, where $sk_j = x_j$. Finally the cloud recovers $rk_{i \rightarrow j} = x_j / x_i \in Z_P$.
- **Sign.** Given private key $sk_i = x_i$, block $m_k \in Z_p$ in shared data M and its block identifier id_k, where $k \in [1, n]$, user u_i outputs the signature on block m_k as $\sigma_k = (H(id_k) w^{m_k})^{x_i} \in G_1$.

- **ReSign.** When user u_i is revoked from the group, the cloud is able to convert signatures of user u_i into signatures of user u_j on the same block. More specifically, given re- signing key $rk_{i\to j}$, public key pk_i , signature σ_k , block m_k and block identifier id_k , the cloud first checks the equation $e(\sigma_k, g) = ?\, e(H(id_k)w^{m_k}, pk_i)$. If the verification result is 0, the cloud outputs $\perp$; otherwise, it outputs $\sigma_k = \sigma_k^{rk_{i\to j}} = (H(id_k)w^{m_k})^{x_i x_j / x_i} = (H(id_k)w^{m_k})^{x_j}$.

  After the re-signing, the original user removes user u_i's *id* from UL and signs the new UL.

- **ProofGen.** To audit the integrity of shared data, the TPA generates an auditing message as follows:

  1) Randomly picks a *c*-element subset L of set [1, *n*] to locate the *c* selected random blocks that will be checked in this auditing task;
  2) Generates a random $y_l \in Z_q$, for $l \in L$ and *q* is a much smaller prime than *p*;
  3) Outputs an auditing message $(l, y_l)_{l\in L}$, and sends it to the cloud.

  After receiving an auditing message, the cloud generates a proof of possession of shared data *M*. More concretely,

  1) The cloud divides set *L* into *d* subset $L_1, \cdots, L_d$, where L_i is the subset of selected blocks signed by user u_i. And the number of elements in subset L_i is c_i. Clearly, we have $c = \sum_{i=1}^{d} c_i$, $L = L_1 \bigcup \cdots \bigcup L_d$ and $L_i \bigcap L_j = \phi$,for $i \neq j$;
  2) For each set L_i, the cloud computes $\alpha_i = \sum_{l\in L_i} y_l m_l \in Z_p$, $\beta_i = \prod_{l\in L_i} \sigma_l^{y_l} \in G_1$ .For L_1, the

cloud computes $\alpha_{11} = \sum_{l \in L_{11}} y_l m_l$, $\beta_{11} = \prod_{l \in L_{11}} \sigma_l^{y_l}$, and $\alpha_{12} = \sum_{l \in L_{12}} y_l m_l$, $\beta_{12} = \prod_{l \in L_{12}} \sigma_l^{y_l}$;

3) Finally, the cloud outputs an auditing proof $(\alpha, \beta, \gamma, id_{l l \in L})$, where $\alpha = (\alpha_1, \cdots, \alpha_d)$, $\beta = (\beta_1, \cdots, \beta_d)$, $\gamma = g^{r_1}$, $\alpha_1 = (\alpha_{11}, \alpha_{12})$, $\beta_1 = (\beta_{11}, \beta_{12})$.

- **ProofVerify.** With an auditing proof $(\alpha, \beta, \gamma, id_{l l \in L})$, an auditing message $(l, y_l)_{l \in L}$,and all the existing users public keys $(pk_1, \cdots, pk_d)$, the TPA checks the correctness of this auditing proof as $e(\prod_{i=1}^{d} \beta_i, g) = \prod_{i=1}^{d} e(\prod_{l \in L_i} H(id_l)^{y_l} \omega^{\alpha_i}, pk_i)$ holds or not. If the result is 1, the verifier believes that the integrity of all the blocks in shared data M is correct. Otherwise, the verifier outputs0.

On the Security of the Protocol

This is section, we will show the weaknesses of Wang et. al's protocol.

Recall that to support dynamic data during public auditing, Wang et al. leverages index hash tables in their public auditing scheme. More precisely, they defined a block identifier as $id_i = \{v_i || r_i || s_i\}$,where $v_i \in N^*$ is the virtual index of this block, r_i is computed as $|id|$ = H`$(m_i || v_i)$ with a collision-resistance hash function $H' : \{0,1\}^* \to Z_q$, and s_i is the signer *id* of block m_i. Note that s_i is used to distinguish the identity of the signer so that the cloud can easily distinguish which block needs to be re-signed during user revocation.

We firstly revisit the execution process of the auditing protocol. After receiving an auditing message, the cloud checks if a user B is revoked or not. If so,the cloud runs the "Resign" algorithm to convert B's signatures into existing group number's signatures, say A. After that, the cloud generates a proof of possession

of shared data M, and sends it to the auditor. Note that $\{id_l\}_{l\in L}$ are in the auditing proof (α, β, $\{id_l\}_{l\in L}$). After receiving this auditing proof, the auditor first distributes each$\{id_l\}_{l\in L}$ to the correct user u_i using s_l in id_l, and then verifies whether the equation holds or not.

Based on the above construction, we will show why the auditor cannot find the correct public key to verify those re-signatures which are converted from the original signatures. More precisely, suppose the cloud tries to convert u_i's signatures into u_j's signatures. In Wang et al.'s scheme the cloud decides which existing group user the signatures should be converted into, thus when the re-sign process is finished, the private key of the converted signatures are changed from x_i to x_j, too. Here comes the problem:

- If the cloud changes the value of s_k, the auditor can find the correct public key pk_j to verity the converted signatures signed by x_j. However, in this case $\{id_l\}_{l\in Li}$ are changed too, and the auditor cannot verify the auditing proof correctly in the ProofVerify stage and the cloud cannot check the message's integrity in the re-sign stage when this data block is resigned again.
- If the cloud does not change the value of s_k, the auditor would not know that the signatures signed by u_i are converted to the signatures signed by u_j. Since $\{id_l\}_{l\in Li}$ are distributed to user u_i using s_i in the tuple of id_i, thus he still use u_i's public key, instead of u_j's public key, to verify the signatures signed by u_j's private key, which would obviously cause to failure.

In Wang et al's paper, they assume that the cloud always converts signatures of a revoked user into the pre-defined original user u_1 who the auditor always knows. However this would cause other problems. For example, the original user is not allowed to revoke from the group, otherwise the problems mentioned above would happen again. For another example, the original user would become a bottleneck and should be on-line if group numbers revoke frequently, since private and authenticated channels among the revoked users, the original user and the cloud are needed when rekeys are computed. Thus always converting

signatures of revoked users into signatures of the original user is not a satisfying solution.

Moreover, in Wang et. al.'s experiments (See Section VI.C), they assume the size of an element of Z_q is $|q|$ = 80 bits, the size of a block identifier is $|id|$ = 80 bits, while we find this is problematic, since the size of $|id|$ must be bigger than the size of r_i and could not be equal where $id_i = \{v_i||r_i||s_i\}$. Thus, we suggest that the size of an element of Z_q is $|q|$=64 bits, the size of a block identifier is $|id|$=96 bits, the size of v_i is 24 bits, and the size of s_i is 8 bits.

Fix on Wang et.al's Public Auditing Mechanism

In this section, we try to fix Wang et al's Public Auditing protocol.

Firstly we check the reason of the weakness. Note that the form of a signature of user i is $(H(id_k)w^{m_k})^{x_A}$ where $H(id_k)$ and m_k are all signed by the private key x_i. Different from other public auditing protocols with dynamic data, to help the cloud to easily distinguish which block needs to be re-signed during user revocation, the signer id of block m_k, i.e., s_i, is added to id_k. When a signature of user u_i is converted into a signature of user u_j using the Resign algorithm, only the private key of u_i is changed to the private key of u_j, while id_k is unchanged. In other words, signer u_i's id (s_k) of the message m_k is unchanged, although now the message is actually signed by user u_j. The inconsistency of the private key and the signer's id contributes the weakness of Wang et al.'s protocol.

Now we give the measurement to fix this weakness. To make the auditor verifies the audit proof correctly, α_i, β_i, $\{id_l\}_{l\in Li}$ and pk_i are used to compute the two sides of the equation. Since $\{id_l\}_{l\in Li}$ are not changed, thus the auditor cannot find the correct public key of the resigner. Our measurement is as follows: When the cloud produces a auditing proof, it also generates a translation list where each record is $u_i \rightarrow u_j$ which means the signatures of u_i is translated to the signature of u_j. After that, the cloud signs this list with its private key, and sends them along with the auditing proof to the auditor. After receiving the auditing proof, the

auditor firstly checks the integrity of the translation list using the cloud's public key. Then it distributes $\{id_l\}_{l \in Li}$ to user u_i by s_i. After that, it checks if $u_i \rightarrow u_j$ is in the list or not. If so, it uses pk_j and $\{id_l\}_{l \in Li}$ } to verify the auditing proof α_i, β_i; otherwise, it uses pk_i and $\{id_l\}_{l \in Li}$ } to verify the auditing proof α_i, β_i.

Since the size of the translation list and its signature is very small, the computation cost for computing the signature by the cloud and verifying the signature by the auditor is also very small. Moreover the communication cost introduced by this list is small, too.

Summary

In this paper, we analyzed a public auditing protocol with efficient user revocation for shared data in the cloud storage proposed in [6], and demonstrated that the auditor cannot find the correct public key to verify those re-signatures which are converted from the original signatures. We also fixed the weakness so that the revoked users' signatures can be verified correctly by the auditor, without losing any features of the original protocol.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (61300172), the Research Fund for the Doctoral Program of Higher Education (20121102120017), the Fund of the State Key Laboratory of Software Development Environment (SKLSDE-2014ZX-14), and the Fundamental Research Funds for the Central Universities（YWF-14-JSJXY-008).

References

[1] B.Wang, B.Li, H.Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," in the Proceedings of ACNS 2012, June 2012, pp.507–525.

[2] G.Ateniese, R.Burns, R.Curtmola, et. al. “Provable Data Possession at Untrusted Stores,” in the Proceedings of ACM CCS 2007, 2007, pp.598–610.

[3] H.Shacham, B.Waters, “Compact Proofs of Retrievability,” in the Proceedings of ASIACRYPT 2008. Springer-Verlag, 2008, pp.90–107.

[4] C.Wang, Q.Wang, K.Ren, W.Lou, “Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing,” in the Proceedings of IEEE INFOCOM 2010, 2010, pp.525–533.

[5] Y.Zhu, H.Wang, Z.Hu, et. al.“Dynamic Audit Services for Integrity Verification of Outsourced Storage in Clouds,” in the Proceedings of ACM SAC 2011, 2011, pp.1550–1557.

[6] B. Wang, B. Li, H. Li," Public auditing for shared data with efficient user revocation in the cloud", INFOCOM 2013, 2904-2912.