Crypto-analyses on “secure and efficient privacy-preserving public auditing scheme for cloud storage”

Yalin Chen[1] and Jue-Sam Chou*[2]

[1] Institute of information systems and applications, National Tsing Hua University

d949702@oz.nthu.edu.tw

[2] Department of Information Management, Nanhua University, Taiwan R.O.C

*: corresponding author: jschou@mail.nhu.edu.tw

Tel: 886+ (0)5+272-1001 ext.56536

Abstract

Recently, Worku et al. pointed out that the work “privacy-preserving public auditing for data storage security in cloud computing” proposed by Wang et al. is insecure and their second work “privacy- preserving public auditing for secure cloud the storage” is inefficient. Thus, they offered a secure and efficient-privacy public auditing scheme for cloud storage. They claimed that their system is provably secure in the random oracle model and the operation is effective. However, after crypto-analysis, we found that the scheme cannot reach the security goal, it has the existential forgery attack. We, therefore, alter it to incorporate the desired privacy preserving requirement, which is very significant in a privacy-preserving public auditing protocol for cloud storage.

1. Introduction

By NIST’s definition, cloud computing has five essential characteristics, three cloud service models, and four cloud deployment models. Besides, cloud security alliance (CSA) has identified multi-tenants as an important element of cloud [1]. From the statement, we can see that cloud computing environments provide human beings many conveniences, whereas they also bring many problems such as, cloud storage security, due to its multi-tenancy nature and the cloud server may itself be un-trustable. In the privacy-preserving public auditing scheme literature, the users don’t possess the outsourced data physically. Hence, checking the integrity of the outsourced encrypted data on the cloud server becomes important. There have been many cryptographic works within this field roughly named privacy-preserving public auditing for cloud storage system designs [2-17]. In 2014, Worku et al. [2] pointed out that Wang et al.s’ work “privacy-preserving public auditing for data storage security in cloud computing” [3] is insecure and their second work “privacy- preserving public auditing for secure cloud the storage” [4] is inefficient. Therefore, they proposed a secure and efficient-privacy public auditing scheme for cloud storage. They claimed that their scheme is provably secure in the random oracle model and the performance is efficient. However, after crypto-analysis, we found that the scheme cannot reach the security goal. It has the existential forgery attack. We, therefore, modify it to comprise the desired requirement, which is very important in a privacy-preserving public auditing protocol for cloud storage. We demonstrate it in this article.

2. Review of Worku et al.'s auditing scheme

Worku et al.'s public auditing for cloud storage design [2], which adopts the framework of an independent third-party auditor (TPA) to audit the outsourced data when needed as does in [3, 4], consists of four basic algorithms; KeyGen, SigGen, ProofGen and VerifyProof. The used notations can be referred to the original article. We briefly describe them below.

2.1 KeyGen

The client generates a random signing key pair *(ssk, spk)*, chooses $x \in_R Z_p$, $u \in_R G$ and computes $v = g^x \in G$. He then uses, *sk =(x, ssk)* as his secret key and *pk =(u, v, g, spk)* as public parameters.

2.2 SigGen

The client chooses a random element in Z_p as the file name $F = \{m_i\}_{1 \le i \le n}$ and computes the file tag t as name||Sigssk(name) with signature on name. Subsequently, for each block $m_i \in Z_p$, the user generates a signature $\sigma_i = (H(i) \cdot u^{m_i})^x \in G (1 \le i \le n)$ and sends to the server for storage. Afterwards, the user deletes the file and its corresponding signatures from local storage. Later, when *TPA* wants to start the auditing protocol, he retrieves the file tag *t* for *F* and checks its validity using *spk*. If the proof of t is correct, the client or *TPA* constructs and sends a challenge *chal* to the server. That is, *TPA* picks random elements c, k_1, k_2 in $\in Z_p$ and sends $chal = (c, k_1, k_2)$ to the server, where k_1, k_2 are randomly chosen as pseudorandom permutation keys by the user for each auditing.

2.3 ProofGen

After receiving *chal*, the server determines the subset $I = \{s_j\}$ $(1 \le j \le c)$ of set [1, n] using pseudorandom permutation $\pi_{key}(\cdot)$ as $S_j = \pi_{k_1}(j)$, and also determines

$v_{s_j} = f_{k_2}(j)(1 \le j \le c)$ using pseudorandom function $f_{key}(.)$. Finally, for $i \in I$, the server computes:

$$\mu^* = \sum_{i=s_1}^{s_c} v_i m_i \ ,$$

$$\sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i}$$

Moreover, the server chooses a random $r \in_R Z_p$ for blinding, using the same function *f*, as $r = f_{k3}(chal)$, where k_3 is a pseudorandom function key generated by the server for each auditing. It then calculates $R = u^r \in G$, computes $\mu = \mu^* + rh(R) \in Z_p$ and sends (μ, σ, R) to *TPA*.

2.4 VerifyProof(*pk*, *chal*)

Upon receiving the proof (μ, σ, R), *TPA* computes $S_j = \pi_{k_1}(j)$, $v_{s_j} = f_{k_2}(j)(1 \le j \le c)$, and verifies the proof by checking Eq. (1) below.

$$e(\sigma, g) ?= e(\prod_{i \in I} H(i)^{v_i} \cdot u^{\mu} \cdot R^{-h(R)}, v) \text{ ....Eq. (1)}$$

The correctness of the verification equation can be shown as follows:

$$e(\sigma, g) = \ e(\prod_{i=s_1}^{s_c} \sigma_i^{v_i}, g) = e(\prod_{i=s_1}^{s_c} (H(i) \cdot u^{m_i})^{x \cdot v_i}, g)$$

$$= e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{v_i m_i}), g)^x \ = e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\sum_{i=s_1}^{s_c} v_i m_i}, g^x)$$

$$= e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\mu^*}, v) = e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\mu - rh(R)}, v)$$

$$= e(\prod_{i \in I} H(i)^{v_i} \cdot u^{\mu} \cdot R^{-h(R)}, v)$$

If Eq. (1) holds, the proof (μ, σ, R) is valid.

3. The weaknesses

For blinding, the server chooses a random element $r \in_R Z_p$, using the same pseudorandom function, as $r = f_{k3}(chal)$, where k_3 is a pseudorandom function key generated by the server for each auditing. It then calculates $R = u^r \in G$ and computes $\mu^* = \sum_{i=s_1}^{s_c} v_i m_i$, $\mu = \mu^* + rh(R) \in Z_p$, and $\sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i}$.

Then, the server sends (, σ, R) to *TPA*.

From the received (μ, σ, R), we can see that since $\sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i} = \prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\sum_{i=s_1}^{s_c} v_i m_i})^x$, a malicious server can regard v_i*s* as constants and m_i*s* as variables. He computes $\mu^* = \sum_{i=s_1}^{s_c} v_i m_i$ using the constants v_i*s* and the message blocks stored. That is, he can obtain an equation containing multiple variables, the m_i*s*, which in mathematics has more than one solution. This means that other than the original m_i*s*, the malicious server can find out the message blocks satisfying the equation without alerting σ. We take S_c=3 as an example. Suppose the values of v_i*s* are (6, 8, 9), and the values of m_i*s* are (1, 4, 2) respectively, then the plan can be defined by $6x + 8y + 9z = 56(= 6m_1^* + 8m_2^* + 9m_3^*)$, where $m_i^*, i=1$ *to 3*, are the forged message blocks. We know that this plane also passes through the point (4, 1, 2). This implies that the malicious server can forge the message blocks from (1, 4, 2) to (4, 1, 2) without alerting the value σ.

Moreover, due to the independence between $\mu^* (= \sum_{i=s_1}^{s_c} v_i m_i)$ and *R*, after intercepting (μ, σ, R), the attacker can set $R' = u^{r'}$ and $\mu' = \mu^* + r' h(R') \in Z_p$ and sends (μ', σ, R') to *TPA*. *TPA* will accept the verification without detection.

4. Modification

From the weaknesses found in section 3, we see that the key point is that the malicious server has the message blocks and the values of v_i*s*. This result in that he can easily find forged message blocks m_i^* to satisfy the value $\mu^*(=\sum_{i=s_1}^{s_c} v_i m_i^*)$ without alerting the value σ. Therefore, we must try to break down the linear structure of value $\mu^*(=\sum_{i=s_1}^{s_c} v_i m_i)$. As a result, we set $\mu^*(=\sum_{i=s_1}^{s_c} v_i m_i h(H(m_i \oplus i)))$ and add the relationship into μ^* and R by setting $\mu = \mu^* + r(h(R) + \mu^*) \in Z_p$ to prevent the found problem. Certainly, we must first let the client's signature σ_i on m_i to be $(H(i) \cdot u^{m_i h(H(m_i \oplus i))})^{x \cdot}$.

Accordingly, if a malicious server launches the above attack on our modification; although, he knows the values of $v_i s$ and $m_i s$, he cannot break the modification. Thus, the privacy is preserved. The correctness of the verification equation can be shown as follows:

$$e(\sigma, g) = e(\prod_{i=s_1}^{s_c} \sigma_i^{v_i}, g) = e(\prod_{i=s_1}^{s_c} (H(i) \cdot u^{m_i h(H(m_i \oplus i))})^{x \cdot v_i}, g)$$

$$= e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{v_i m_i h(H(m_i \oplus i))}), g)^x = e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\sum_{i=s_1}^{s_c} v_i m_i h(H(m_i \oplus i))}, g^x)$$

$$= e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\mu^*}, v) = e(\prod_{i=s_1}^{s_c} (H(i)^{v_i} \cdot u^{\mu - r(h(R)+\mu^*)}, v)$$

$$= e(\prod_{i \in I} H(i)^{v_i} \cdot u^{\mu} \cdot R^{-(h(R)+\mu^*)}, v)$$

5. Conclusion

In this paper, we showed that Worku et al.'s work privacy-preserving public auditing for data storage security in cloud computing is flawed. It suffers from the existential forgery attack. For enhancing its security, we therefore modified it to avoid the weaknesses. From the analysis shown in section 4, we see that we have reached the goal of the security promotion.

References

[1] CSA, *security guidance for critical areas of focus in cloud computing V3.0*, Cloud Security Alliance, 2011

[2] Worku, S. G., Xu, C., Zhao, J., & He, X. Secure and efficient privacy-preserving

public auditing scheme for cloud storage. Computers & Electrical Engineering 40, (2014), 1703-1713.

[3] Wang C,Wang Q, Ren K, Lou W. Privacy-preserving public auditing for data storage security in cloud computing. In: INFOCOM, 2010 proceedings IEEE; 2010. p. 1–9.

[4] Wang C, Chow S, Wang Q, Ren K, Lou W. Privacy-preserving public auditing for secure cloud storage. IEEE Trans Comput 2011

[5] Ateniese G, Burns R, Curtmola R, Herring J, Khan O, Kissner L, et al. Remote data checking using provable data possession. ACM Trans Inf Syst Secur2011;14:1–34.

[6] Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, et al. Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on computer and communications security, CCS 2007. p. 598–609.

[7] Bowers KD, Juels A, Oprea A. HAIL: a high-availability and integrity layer for cloud storage. In: Proceedings of the 16th ACM conference on computer and communications security, CCS 2009. p. 187–98.

[8] Deswarte Y, Quisquater J-J, Saïdane A. In: Jajodia S, Strous L, editors. Remote integrity checking: integrity and internal control in information systems VI, vol. 140. Boston: Springer; 2004. p. 1–11.

[9] Dodis Y, Vadhan S, Wichs D. Proofs of retrievability via hardness amplification. In: Reingold O, editor. Theory of cryptography, vol. 5444. Berlin/Heidelberg: Springer; 2009. p. 109–27.

[10] Erway C, Kupcu A, Papamanthou C, Tamassia R. Dynamic provable data possession. In: Proceedings of the 16th ACM conference on computer and communications security, CCS 2009. p. 213–22.

[11] Shacham H, Waters B. Compact proofs of retrievability. In: Pieprzyk J, editor. Advances in cryptology – ASIACRYPT 2008, vol. 5350. Berlin/Heidelberg: Springer; 2008. p. 90–107.

[12] Zheng Q, Xu S. Secure and efficient proof of storage with deduplication. In: Proceedings of the second ACM conference on data and application security and privacy, CODASPY 2012. p. 1–12.

[13] Zheng Q, Xu S. Fair and dynamic proofs of retrievability. In: Proceedings of the first ACM conference on data and application security and privacy, CODASPY 2011. p. 237–48.

[14] Wang Q, Wang C, Li J, Ren K, Lou W. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes M, Ning P, editors. Computer security – ESORICS 2009, vol. 5789. Berlin/Heidelberg: Springer; 2009. p. 355–70.

[15] Zhuo H, Sheng Z, Nenghai Y. A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans Knowl Data Eng 2011;23:1432–7.
[16] Chunxiang X, Xiaohu H, Daniel-Abraha W. Cryptanalysis of Wang's auditing protocol for data storage security in cloud computing. In: Liu C et al., editors. Information computing and applications, vol. 308. Berlin Heidelberg: Springer; 2012. p. 422–8.
[17] Wang Q,Wang C, Ren K, LouW, Li J. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans Parallel Distrib Syst 2011;22:847–59.