On the Security of a Cloud Data Storage Auditing Protocol IPAD

Xu An Wang[1,2](✉), Xiaoshuang Luo[1], Jindan Zhang[3], and Xiaoyuan Yang[1]

[1] Key Laboratory of Cryptology and Information Security, Engineering University of CAPF, Xi'an 710086, China
wangxazjd@163.com

[2] Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, People's Republic of China

[3] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

Abstract. Nowadays cloud data storage is a very important storage service for us, but to ensure the datum stored in the remote cloud server remains unmodified, we need a mechanism to check the datum's integrity, cloud data storage auditing protocol is such a mechanism, which has received great attention from researchers. Recently Zhang et al. proposed an efficient ID-based public auditing protocol called IPAD for the outsourced data by combing Waters signature and public auditing for the outsourced data. They claimed IPAD is the first ID-based auditing protocol for data integrity in the standard security model. But in this paper we show their proposal is not secure. Especially, the adversaries can easily generate tags for any file, which obviously break the unforgeability property of the cloud storage auditing protocol.

1 Introduction

In these days, cloud computation is a very hot research topic for its promising properties of cheap management cost for users, any where/any time access, and very scalable software and hard ware investigation [21]. However, before adapting cloud computation, data owners should ensure their data shall be secure and well protected [22–24]. Integrity is one of the most important security property for cloud storage. However when the data owners outsource their datum to the cloud, the datum is not controlled by the owners any more, how to ensure the datum has not been modified and changed? Cloud storage auditing protocol is such a mechanism. In 2007, the first provable data possession (PDP) scheme was proposed by Atenesis et al. [1,2]. Also the proof of retrievability protocol for cloud storage was proposed by Jules et al. [3] and Shacham and Waters [4]. Later many cloud auditing protocols with different properties have been proposed, such as cloud auditing protocols with dynamic updates [5–9], cloud auditing protocols with publicly verifiability [4,11], cloud auditing protocols with privacy-preserving [11–13], cloud auditing protocols with other interesting properties [14–18].

© Springer International Publishing AG 2018
L. Barolli et al. (eds.), *Advances in Internetworking, Data & Web Technologies*,
Lecture Notes on Data Engineering and Communications Technologies 6,
DOI 10.1007/978-3-319-59463-7_61

Recently, Zhang et al. [25] proposed an efficient ID-based public auditing protocol called IPAD for the outsourced data by combing Waters signature and public auditing for the outsourced data. They claimed IPAD is the first ID-based auditing protocol for data integrity in the standard security model. But in this paper we show their proposal is not secure. Especially, the adversaries can easily generate tags for any file, which obviously break the unforgeability property of the cloud storage auditing protocol.

2 Review of Zhang et al.'s IPAD Scheme

Here we first review Zhang et al.'s ID-based public auditing protocol in the standard model [25]. It consists the following six algorithms: Setup, Key-extract, TagGen, Challenge, Proof, Verifying, the details are given as follows:

1. Setup. For public key generator (PKG), it sets up the following system parameters. Given a security parameter k, it selects two multiplicative cyclic groups $(\mathbb{G}_1, \mathbb{G}_2)$ of prime order $q \geq 2^k$, let $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a bilinear map and $g \in_R \mathbb{G}_1$ be a generator of group $\mathbb{G}_1$. h, g_2 are two random generators of group $\mathbb{G}_1$. Let $H_1 : \{0,1\}^* \rightarrow Z_q$ and $H_v : \{0,1\}^* \rightarrow \{0,1\}^{n_v}$ be two collision-resistant hash functions, where $n_v \in Z$. And randomly choose $\alpha \in Z_q$ as master key of PKG and compute the corresponding public key $P_{pub} = g^{\alpha}$. Also randomly choose the following elements:
   a. $v' \in_R \mathbb{G}_1$
   b. $v_i \in_R \mathbb{G}_1$ for $i = 1, \cdots, n_v$. Let $V = \{v_i\}_{i\ in 1, \cdots, n_v}$

   Finally publish the following system parameters:

   $$Param = (\mathbb{G}_1, \mathbb{G}_2, q, e, v', V, g, h, H_1, H_v, g_2, P_{pub})$$

   At the same time, PKG secretly keeps his master secret key s.
2. Key Extraction. For a data user with identity ID_j, if it wants to register his identity ID_j to PKG, the following steps are executed:
   a. First it submits his identity ID to the PKG.
   b. PKG computes $\mathfrak{B}_j = H_v(ID_j)$. Let $\mathfrak{B}_j[i]$ be the i-th bit of $\mathfrak{B}_j$. Then define $V_j \subset \{1, \cdots, n_v\}$ be the set of indicies such that $\mathfrak{B}_j = 1$
   c. To produce the private key d_j of the data user with identity ID_j, PKG randomly choose $a_{u_j} \in_R Z_q$ to compute

   $$d_j = (d_{j1}, d_{j2}) = (g_2^{\alpha}(v' \prod_{i \in V_j} v_i)^{a_{u_j}}, g^{a_{u_j}})$$
3. TagGen. Given a data file M, the data user with identity ID_j splits M into n blocks such that each block has s sectors. Namely, $M = m_1 || \cdots || m_n$ and $m_i = m_{i1} || \cdots || m_{is}$. Then it chooses a random file name $Name$ from a sufficiently large domain Z_q^* and $s+1$ random values $r_0, r_1, \cdots, r_s \in_R Z_q$ to compute $u_i = g_2^{r}$ for each $0 \leq i \leq s$.

To produce file tag, it also does the following steps:

a. Compute $(pk_s, sk_s) \leftarrow \Sigma.KeyGen(1^k)$ to obtain a pair of public/private keys, where Σ is a secure signature algorithm.

b. Compute $\Phi = \Sigma.sign(sk_s, \tau_0)$ to obtain a signature on string τ_0, where $\tau_0 =$ "$Name||n||u_0||u_1||\cdots||u_s$"

c. For each data block m_i, $1 \leq i \leq n$, it computes

$$\omega_i = r_0 H_1(Name||i) + \sum_{j=1}^{s} r_j m_{ij}$$

d. The authentication tag on data block m is computed as

$$t_i = \ (t_{i,1} = (d_{j1})^{\omega_i} = g_2^{\varepsilon_1}(v' \prod_{i \in V_j} v_i)^{\varepsilon_2},$$
$$t_{i,2} = d_{j2}^{\omega_i} = g^{\varepsilon_2})$$

where $\varepsilon = \alpha \cdot \omega_i$, $\varepsilon = a_{u_j} \cdot \omega_i$ (Note that to produce a probabilistic signature, the data user with identity ID_j also can select $\hat{r} \in Z_q$ to compute

$$t_i = \ (t_{i1} = (d_{j1})^{\delta_i}(v' \prod_{i \in V_j} v_i)^{\hat{r}},$$
$$t_{i2} = d_{j2}^{\delta_i} \cdot g^{\hat{r}})$$

Finally, the data user sends the data file M together with all the authentication tag t_i, $1 \leq i \leq n$ to the cloud storage server, And delete the above random values $r_0, r_1, \cdots, r_s$, private key sk_s of signature algorithm $\sum$ and the local file M.

4. Challenge Phase. To check data integrity of the outsourced data, the auditor first verifies whether the signature ϕ is valid by invoking $\sigma.Verify(\phi, pk_s)$. If it is not, outputs 0 and terminates it. otherwise, the auditor parses τ to recover the file, name $Name$ and n as well as $u_0, u_1, \cdots, u_s$. Then it randomly chooses a l-element subset I of the set $[1, n]$ and a number $\rho \in Z_q$ to produce the following challenging message

$$Chall = \{\rho, I\}$$

and sends them the cloud storage server.

5. Prove. Upon receiving the challenging message $Chall = (I, \rho)$, the cloud storage server first produces a l-element set $Q = (i, \beta_i)$ where $i \in I, \beta_i = \rho^i \bmod q$, Then based on the outsourced data file $M = \{m_1, \cdots, m_n\}$ and authentication tags t_i, $1 \leq i \leq n$, it computes

$$\delta_1 = \prod_{i \in I} t_{i1}^{\beta_i}$$

$$\delta_2 = \prod_{i \in I} t_{i2}^{\beta_i}$$

and for $j = 1$ to s, it computes

$$\mu_j = \sum_{i \in I} \beta_i m_{ij}$$

Finally, the cloud storage server responds the auditor with the corresponding proof information $Prf = (\delta_1, \delta_2, \{\mu_j\}_{j=1,\cdots,s})$

6. Verifying. According to the responded proof information $Prf = (\delta_1, \delta_2, \{\mu_j\}_{j=1,\cdots,s})$, the auditor first computes

$$\hat{h} = \sum_{i \in I} \beta_i H_1(Name||i)$$

Then it verifies the integrity of data file by the following equation

$$e(u_0^{\hat{h}} \cdot \prod_{i=1}^{s} u_i^{\mu_j}, P_{pub}) e(v' \cdot \prod_{i \in V_j} v_i, \delta_2) = e(\delta_1, g)$$

If the above Equation holds, the auditor outputs $VerifyRst$ as accept; otherwise, output $VerifyRSt$ as reject.

3 Our Attack

Our attack shows that the adversary can forge tags for any new files.

- For the deterministic tag generation algorithm, the attack runs as the following:
  1. First the adversary can query on the data owner ID_t for the tag generation on different data blocks $(m_1, m_2, \cdots, m_n)$, he can get the following tags for $(m_1, m_2, \cdots, m_n)$:

$$\begin{aligned} t_{11} &= (d_{t1})^{r_0 H_1(Name||1) + \sum_{j=1}^{s} r_j m_{1j}}, \\ t_{12} &= (d_{t2})^{r_0 H_1(Name||1) + \sum_{j=1}^{s} r_j m_{1j}}, \\ &\cdots\cdots\cdots \\ t_{n1} &= (d_{t1})^{r_0 H_1(Name||n) + \sum_{j=1}^{s} r_j m_{nj}}, \\ t_{n2} &= (d_{t2})^{r_0 H_1(Name||n) + \sum_{j=1}^{s} r_j m_{nj}}, \end{aligned}$$

  2. Let $A_j = (d_{t1})^{r_j} (0 \le j \le n)$, the adversary can get the following equations:

$$t_{11} = (d_{t1})^{r_0 H_1(Name||1) + \sum_{j=1}^{s} r_j m_{1j}} = (A_0)^{H_1(Name||1)} (A_1)^{m_{11}} \cdots (A_s)^{m_{1s}} \quad (1)$$

$$\cdots\cdots\cdots$$

$$t_{n1} = (d_{t1})^{r_0 H_1(Name||n) + \sum_{j=1}^{s} r_j m_{nj}} = (A_0)^{H_1(Name||n)} (A_1)^{m_{n1}} \cdots (A_s)^{m_{ns}} \quad (\text{n})$$

3. Note in the above equations, $H_1(Name||1), \cdots, H_1(Name||n), m_{11}, \cdots, m_{1s}, m_{n1}, \cdots, m_{ns}$ are all known to anyone including the adversary, thus he can compute

$$A_0, A_1, \cdots, A_s$$

by implementing linear transformation on the exponentials with high probability (in case the computation fails, the adversary can query on new files with different data blocks and compute again until succeeding).

4. Once the adversary get $A_0, A_1, \cdots, A_s$, he can compute tags for any file with name $Name*$ and $m^*_{11}, \cdots, m^*_{1s}, m^*_{n1}, \cdots, m^*_{ns}$ as following:

$$\begin{aligned}
t_{11} &= (d_{t1})^{r_0 H_1(Name^*||1)+\sum_{j=1}^{s} r_j m^*_{1j}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{11}} \cdots (A_s)^{m^*_{1s}} \\
t_{12} &= (d_{t2})^{r_0 H_1(Name^*||1)+\sum_{j=1}^{s} r_j m^*_{1j}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{11}} \cdots (A_s)^{m^*_{1s}} \\
&\qquad \cdots\cdots\cdots \\
t_{n1} &= (d_{t1})^{r_0 H_1(Name^*||n)+\sum_{j=1}^{s} r_j m^*_{nj}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{n1}} \cdots (A_s)^{m^*_{ns}} \\
t_{n2} &= (d_{t2})^{r_0 H_1(Name^*||n)+\sum_{j=1}^{s} r_j m^*_{nj}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{n1}} \cdots (A_s)^{m^*_{ns}}
\end{aligned}$$

5. Thus the adversary could generate tags for any file, which obviously break the un-forgeability property of the cloud storage auditing protocol.

- For the randomized tag generation algorithm, the attack runs as the following:
  1. First the adversary can query on the data owner ID_t for the tag generation on different data blocks $(m_1, m_2, \cdots, m_n)$, he can get the following tags for $(m_1, m_2, \cdots, m_n)$:

$$\begin{aligned}
t_{11} &= (d_{t1})^{r_0 H_1(Name||1)+\sum_{j=1}^{s} r_j m_{1j}}, \\
t_{12} &= (d_{t2})^{r_0 H_1(Name||1)+\sum_{j=1}^{s} r_j m_{1j}}, \\
&\qquad \cdots\cdots\cdots \\
t_{n1} &= (d_{t1})^{r_0 H_1(Name||n)+\sum_{j=1}^{s} r_j m_{nj}}, \\
t_{n2} &= (d_{t2})^{r_0 H_1(Name||n)+\sum_{j=1}^{s} r_j m_{nj}},
\end{aligned}$$

  2. Let $A_j = (d_{t1})^{r_j} (0 \le j \le n)$, the adversary can get the following equations:

$$t_{11} = (d_{t1})^{r_0 H_1(Name||1)+\sum_{j=1}^{s} r_j m_{1j}} = (A_0)^{H_1(Name||1)}(A_1)^{m_{11}} \cdots (A_s)^{m_{1s}} \quad (1)$$

$$\cdots\cdots\cdots$$

$$t_{n1} = (d_{t1})^{r_0 H_1(Name||n)+\sum_{j=1}^{s} r_j m_{nj}} = (A_0)^{H_1(Name||n)}(A_1)^{m_{n1}} \cdots (A_s)^{m_{ns}} \quad (\text{n})$$

  3. Note in the above equations, $H_1(Name||1), \cdots, H_1(Name||n), m_{11}, \cdots, m_{1s}, m_{n1}, \cdots, m_{ns}$ are all known to anyone including the adversary, thus he can compute

$$A_0, A_1, \cdots, A_s$$

by implementing linear transformation on the exponentials with high probability (in case the computation fails, the adversary can query on new files with different data blocks and compute again until succeeding).

4. Once the adversary get $A_0, A_1, \cdots, A_s$, he can compute tags for any file with name $Name*$ and $m^*_{11}, \cdots, m^*_{1s}, m^*_{n1}, \cdots, m^*_{ns}$ as following:

$$
\begin{aligned}
t_{11} &= (d_{t1})^{r_0 H_1(Name^*||1) + \sum_{j=1}^{s} r_j m^*_{1j}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{11}} \cdots (A_s)^{m^*_{1s}} \\
t_{12} &= (d_{t2})^{r_0 H_1(Name^*||1) + \sum_{j=1}^{s} r_j m^*_{1j}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{11}} \cdots (A_s)^{m^*_{1s}} \\
&\cdots\cdots\cdots \\
t_{n1} &= (d_{t1})^{r_0 H_1(Name^*||n) + \sum_{j=1}^{s} r_j m^*_{nj}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{n1}} \cdots (A_s)^{m^*_{ns}} \\
t_{n2} &= (d_{t2})^{r_0 H_1(Name^*||n) + \sum_{j=1}^{s} r_j m^*_{nj}} = (A_0)^{H_1(Name^*||1)}(A_1)^{m^*_{n1}} \cdots (A_s)^{m^*_{ns}}
\end{aligned}
$$

5. Thus the adversary could generate tags for any file, which obviously break the un-forgeability property of the cloud storage auditing protocol.

4 Conclusion

In this paper, we show a recent proposed cloud auditing protocol is not secure, the reason why their scheme is not secure is that, the tag generation algorithm is not secure, by querying many times of tag generation oracle, the adversary can easily forge new tags for any block. We point out this attack is a very basic result, we leave how to strengthen their scheme to be secure as our future work.

Acknowledgements. This work is supported by Nature Science Foundation of China (Grant NO. U1636114, 61572521) and Natural Science Foundation of Shaanxi Province (Grant No. 2016JQ6037) and Guangxi Key Laboratory of Cryptography and Information Security (Grant No. GCIS201610).

References

1. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.: Provable data possession at untrusted stores. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS 2007, pp. 598–609. ACM Press, Alexandria (2007)
2. Ateniese, G., Burns, R.C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.: Remote data checking using provable data possession. ACM Trans. Inf. Syst. Secur. **14**(1), 12 (2011)
3. Juels, A., Kaliski Jr, B.S.: PORS: proofs of retrievability for large files. In: Ning, P., di Vimercati S.D.C., Syverson P.F. (eds.) ACM CCS 2007, pp. 584–597. ACM Press, Alexandria (2007)
4. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
5. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: Sadeghi, A.R., Gligor, V.D., Yung, M. (eds.) ACM CCS 2013, pp. 325–336. ACM Press, Berlin (2013)
6. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 279–295. Springer, Berlin (2013)

7. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**(5), 847–859 (2012)
8. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distrib. Syst. **24**(9), 1717–1726 (2013)
9. Wang, B., Baochun, L., Hui, L.: Public auditing for shared data with efficient user revocation in the cloud. In: Proceedings of the 33th Conference on Information Communications (INFOCOM 2013), pp. 2750–2758. IEEE Press (2013)
10. Yuan, J., Yu, S.: Proofs of retrievability with public verifiability and constant communication cost in cloud. In: Proceedings of the 2013 International Workshop on Security in Cloud Computing, Cloud Computing, pp. 19–26 (2013)
11. Wang, C., Chow, S., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. **62**(2), 362–375 (2013)
12. Yu, Y., Zhang, Y., Ni, J., Au, M., Chen, L., Liu, H.: Remote data possession checking with enhanced security for cloud storage. Future Gener. Comput. Syst. **52**, 77–85 (2014). doi:10.1016/j.future.2014.10.006
13. Yu, Y., Au, M.H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., Min, G.: Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. Inf. Forensics Secur. **12**(4), 767–778 (2016). doi:10.1109/TIFS.2016.2615853
14. Zhu, Y., Hu, H., Ahn, G., Yu, M.: Cooperative provable data possession for integrity verification in multi cloud storage. IEEE Trans. Parallel Distrib. Syst. **23**(12), 2231–2244 (2012)
15. Halevi, S., Harnik, D., Pinkas, B., Shulman-Peleg, A.: Proofs of ownership in remote storage systems. In: Chen, Y., Danezis, G., Shmatikov, V. (eds.) ACM CCS 2011, pp. 491–500. ACM Press, Chicago (2011)
16. Zheng, Q., Xu, S.: Secure and efficient proof of storage with deduplication. Cryptology ePrint Archive, Report 2011/529 (2011). http://eprint.iacr.org/2011/529
17. Yuan, J., Yu, S.: Public integrity auditing for dynamic data sharing with multi-user modification. IEEE Trans. Inf. Forensics Secur. **10**(8), 1717–1726 (2015)
18. Yu, Y., Li, Y., Ni, J., Yang, G., Mu, Y., Susilo, W.: Comments on "public integrity auditing for dynamic data sharing with multi-user modification". IEEE Trans. Inf. Forensics Secur. **11**(3), 658–659 (2016)
19. Yuan, J., Yu, S.: PCPOR: public and constant-cost proofs of retrievability in cloud. J. Comput. Secur. **23**, 403–425 (2015)
20. Yuan, J., Yu, S.: Efficient public integrity checking for cloud data sharing with multi-user modification. In: Proceedings of the 33rd Conference on Information Communications (INFOCOM 2014), pp. 2121–2129. IEEE Press (2014)
21. Puzar, M., Plagemann, T.: Data sharing in mobile ad-hoc networks-a study of replication and performance in the MIDAS data space. Int. J. Space-Based Situated Comput. **1**(2/3), 137–150 (2015)
22. Petrlic, R., Sekula, S., Sorge, C.: A privacy-friendly architecture for future cloud computing. Int. J. Grid Util. Comput. **4**(4), 265–277 (2013)
23. Wang, Y., Du, J., Cheng, X., Liu, Z., Lin, K.: Degradation and encryption for outsourced PNG images in cloud storage. Int. J. Grid Util. Comput. **7**(1), 22–28 (2016)
24. Ye, X., Khoussainov, B.: Fine-grained access control for cloud computing. Int. J. Grid Util. Comput. **4**(2/3), 160–168 (2013)
25. Zhang, J., Li, P., Mao, J.: IPad: ID-based public auditing for the outsourced data in the standard model. Cluster Comput. **19**(1), 127–138 (2016). doi:10.1007/s10586-015-0511-3