Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Comment:

Cryptanalysis of an identity-based public auditing protocol for cloud storage*

Li-bing WU[1], Jing WANG[1], De-biao HE[‡2], Muhammad-Khurram KHAN[3]

([1]*School of Computer Science, Wuhan University, Wuhan 430072, China*)

([2]*School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China*)

([3]*Center of Excellence in Information Assurance (CoEIA), King Saud University, Riyadh 11653, Saudi Arabia*)

E-mail: whuwlb@126.com; cswjing@whu.edu.cn; hedebiao@163.com; mkhurram@ksu.edu.sa

Received Sept. 6, 2016; Revision accepted Jan. 23, 2017; Crosschecked Dec. 20, 2017

Abstract: Public verification of data integrity is crucial for promoting the serviceability of cloud storage systems. Recently, Tan and Jia (2014) proposed an identity-based public verification (NaEPASC) protocol for cloud data to simplify key management and alleviate the burden of check tasks. They claimed that NaEPASC enables a third-party auditor (TPA) to verify the integrity of outsourced data with high efficiency and security in a cloud computing environment. However, in this paper, we pinpoint that NaEPASC is vulnerable to the signature forgery attack in the setup phase; i.e., a malicious cloud server can forge a valid signature for an arbitrary data block by using two correct signatures. Moreover, we demonstrate that NaEPASC is subject to data privacy threats in the challenge phase; i.e., an external attacker acting as a TPA can reveal the content of outsourced data. The analysis shows that NaEPASC is not secure in the data verification process. Therefore, our work is helpful for cryptographers and engineers to design and implement more secure and efficient identity-based public auditing schemes for cloud storage.

Key words: Cloud data; Public auditing; Data integrity; Data privacy
https://doi.org/10.1631/FITEE.1601530 **CLC number:** TP309

1 Introduction

With the explosive growth of data in the world, cloud storage now plays an increasingly important role in storing and managing massive data. The traits of powerful storage, high reliability, flexible access, and affordable management bring a sense of convenience to both individuals and business organizations (Li *et al.*, 2016; Liu *et al.*, 2016). By outsourcing their data to a remote cloud server, individuals and business organizations free themselves from the management of unexpected system failures. However, shifting data from local storage to the cloud entails some security and privacy challenges owing to the loss of data ownership. Hence, maintaining the integrity of outsourced data is a key issue in the serviceability of cloud storage.

To address the security issue, many encryption protocols (Fu *et al.*, 2015; 2016; Xia *et al.*, 2016) and public auditing protocols (Guo *et al.*, 2014; He *et al.*, 2015; Ren *et al.*, 2015) have been proposed to check the storage integrity of cloud data without downloading the whole file. Ateniese *et al.* (2007) first proposed a public auditing mechanism, called 'provable data possession (PDP)', to verify the data integrity on remote nodes. Based on the mechanism from Ateniese *et al.* (2007), Shacham and Waters (2008; 2013) came up with an improved PDP scheme

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61472287, 61501333, 61572379, and 61772377), the Natural Science Foundation of Hubei Province, China (Nos. 2015CFA068 and 2017CFA007), the Wuhan Science and Technology Plan Project (No. 2016060101010047), and the Deanship of Scientific Research at King Saud University, Saudi Arabia (No. PRG-1436-16)

ORCID: De-biao HE, http://orcid.org/0000-0002-2446-7436

with a Boneh–Lynn–Shacham (BLS) signature. The construction of dynamic verification of data was first presented by Chen and Curtmola (2012), which used error-correcting codes to support provable updates on outsourced data files. However, these schemes ignore certain characteristics of cloud users.

In some other similar studies, Wang *et al.* (2013) designed another public auditing scheme to preserve data privacy. It prevents the TPA from disclosing the outsourced data using a masking technique. Nonetheless, Tan and Jia (2014) claimed that the scheme and some other work are suitable only for one situation: one key, one file. Once the key is lost, the cloud user would no longer be able to verify the data integrity. Furthermore, the cloud user needs to remember different keys for various data files if he/she needs to outsource multiple data files into the cloud. Otherwise, the cloud server could forge the metadata of each data block to deceive the user if his/her key pairs are reused for different files.

To overcome the problem noted above, Tan and Jia (2014) proposed an identity-based public verification protocol (NaEPASC), which simplifies key management and alleviates the user burden. They asserted that NaEPASC is secure and efficient in auditing the integrity of the outsourced data. However, our cryptanalysis shows that their NaEPASC protocol is not secure when checking the data integrity for cloud storage, because a malicious cloud server can modify (or delete) the data block without being detected by the TPA. NaEPASC is vulnerable to the data privacy attack; i.e., an external attacker could disclose the data information by acting as the TPA. Hence, our work will be able to help cryptographers and engineers design more secure and efficient identity-based public auditing schemes for cloud storage by avoiding these two weaknesses.

2 Review of NaEPASC

In this section, we give a brief overview of the construction of the NaEPASC protocol (Tan and Jia, 2014). Tan and Jia (2014) divided the auditing process of their NaEPASC protocol into two phases: setup phase and challenge phase. Here, some definitions are presented: G and G_{T} denote two cyclic groups of prime order p, and $e : G \times G \rightarrow G_{\mathrm{T}}$ is a bilinear map. $H_1, H_2 : \{0,1\}^* \rightarrow G$ and $H_3 : \{0,1\}^* \rightarrow Z_p$ refer to three types of cryptographic hash functions.

1. Setup phase. The PKG first executes the algorithm KeyGen to select a random $x \in Z_p$ as its secret parameter, and set (P, Q) as the public parameter, where $P \in G$ is an arbitrary generator and $Q = xP$. Then PKG sends the secret key $\{xP_j\}_{j=0,1}$ and the public key (P, Q) to the cloud user, where $P_j = H_1(\mathrm{ID}, j)$ and ID is the identity of the cloud user.

Suppose that the data file F named 'filename' is divided into n blocks, and id_i is the index of data block m_i. Then, the cloud user runs the algorithm Sign to generate the signatures as noted below.

For $1 \leq i \leq n$, the user first selects a random element $r_i \in Z_p$ to compute the value of $T_i = r_i P$. Next, the user computes $S_i = r_i P_{\mathrm{w}} + c_i x P_0 + m_i x P_1$, where $P_{\mathrm{w}} = H_2(\mathrm{filename}), c_i = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_i)$. Finally, the user sends $\{\mathrm{filename}, \{S_i, T_i\}_{1 \leq i \leq n}\}$ to the cloud server and removes it from local storage.

2. Challenge phase. First, the TPA randomly chooses a c-element subset $I = \{l_1, l_2, \cdots, l_c\}$ of the set $\{1, 2, \cdots, n\}$. Then, it sets $\mathrm{chal} = \{i, v_i\}_{i \in I}$ as the auditing challenge and sends it to the cloud server, where v_i is randomly chosen from the group Z_q, $|q| = |p|/2$.

Upon receiving the message $\mathrm{chal} = \{i, v_i\}_{i \in I}$, the cloud server executes the algorithm GenProof to compute the corresponding proof:

$$\begin{cases} \mu = \sum_{i \in I} v_i m_i, \\ S_n = \sum_{i \in I} v_i S_i, \\ T_n = \sum_{i \in I} v_i T_i. \end{cases} \tag{1}$$

Next, it sends the proof $\{\mu, S_n, T_n\}$ to the TPA as the response.

Finally, the TPA performs VerifyProof to check the data integrity using the following equation:

$$e(S_n, P) = e(T_n, P_{\mathrm{w}}) e\left(\sum_{l_1}^{l_c} c_i v_i P_0 + \mu P_1, Q\right), \tag{2}$$

where $c_i = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_i)$, $P_0 = H_1(\mathrm{ID}, 0)$, $P_1 = H_1(\mathrm{ID}, 1)$, and $P_{\mathrm{w}} = H_2(\mathrm{filename})$.

3 Cryptanalysis of NaEPASC

Tan and Jia (2014) claimed that NaEPASC protocol is efficient and secure in verifying the integrity of outsourced data. In this section, we will analyze the security of the NaEPASC protocol and point out two concrete attacks against NaEPASC. In the signature forgery attack, a malicious cloud server can modify (or delete) the outsourced data without being detected by the TPA. In the data recovery attack, a curious TPA can recover the content of the outsourced data from auditing messages. The detailed attacks are presented below.

3.1 Signature forgery attack

Tan and Jia (2014) proved that their signature scheme is existentially unforgeable in Theorem 2. However, we demonstrate that it is vulnerable to a signature forgery attack, because a malicious cloud is able to forge a valid signature for an arbitrary data block. Consequently, the malicious cloud server can modify (or delete) the original user's data and pass the TPA's verification successfully.

Suppose a data file F is split into n blocks, i.e., $F = m_1 \parallel m_2 \parallel \cdots \parallel m_n$, and the signature of each data block m_i is denoted as $\sigma_i = \{S_i, T_i\}$. Let $\mathscr{A}_1$ denote the malicious cloud server. It can commit the signature forgery attack through the following steps:

1. $\mathscr{A}_1$ extracts two correctly stored data blocks m_j and m_k and their corresponding signatures $\{S_j, T_j\}$ and $\{S_k, T_k\}$, where $j, k \in \{1, 2, \cdots, n\}$.

2. $\mathscr{A}_1$ computes $c_j = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_j)$ and $c_k = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_k)$ with the known parameters $H_3, \mathrm{ID}, \mathrm{filename}, \mathrm{id}_j$, and id_k.

3. For each i, $\mathscr{A}_1$ computes the value of $c_i = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_i)$ and stores it $(i \neq k, j)$.

4. For each $m_i' \neq m_i$, $\mathscr{A}_1$ computes $T_i' = a_iT_j + b_iT_k$ and $S_i' = a_iS_j + b_iS_k, (i \neq k, j)$, where

$$\begin{cases} a_i = \dfrac{c_jm_i' - c_im_j}{c_jm_k - c_km_j}, \\ b_i = \dfrac{c_km_i' - c_im_k}{c_km_j - c_jm_k}. \end{cases} \tag{3}$$

5. $\mathscr{A}_1$ substitutes m_i', T_i', and S_i' for m_i, T_i, and S_i $(i \neq k, j)$.

6. In the same way, $\mathscr{A}_1$ can replace $\{m_j, T_j, S_j\}$ and $\{m_k, T_k, S_k\}$ with $\{m_j', T_j', S_j'\}$ and $\{m_k', T_k', S_k'\}$, respectively.

7. Upon receiving the auditing challenge $\mathrm{chal} = \{(i, v_i)\}_{i\in I}$ from the TPA, $\mathscr{A}_1$ computes $\mu' = \sum_{i\in I} v_im_i', S_n' = \sum_{i\in I} v_iS_i', T_n' = \sum_{i\in I} v_iT_i'$, where $I = \{l_1, l_2, \cdots, l_c\}$ is a subset from the set $\{1, 2, \cdots, n\}$.

8. $\mathscr{A}_1$ returns $\{\mu', S_n', T_n'\}$ as the auditing proof.

$\mathscr{A}_1$'s response can pass the TPA's verification. We present the proof below:

Owing to $T_i' = a_iT_j + b_iT_k = (a_ir_j + b_ir_k)P$, $S_i' = a_iS_j + b_iS_k = (a_ir_j + b_ir_k)P_\mathrm{w} + c_ixP_0 + m'xP_1$ (where a, b are the values computed by $\mathscr{A}_1$ in step 4), we can obtain Eq. (4).

Thus, the proof $\{\mu', S_n', T_n'\}$ can pass the verification and modify the outsourced data without being detected by the TPA.

In a special data modification case, the malicious cloud server can preserve only two legitimate and authentic pairs of the data blocks and their signatures, and then deletes all other blocks to save space. During the challenge phase, it can forge a series of data blocks and corresponding signatures

$$\begin{aligned} e(S_n', P) =& e\left(\sum_{l_1}^{l_c} v_iS_i', P\right) = e\left[\sum_{l_1}^{l_c} v_i(a_iS_j + b_iS_k), P\right] \\ =& e\left(\sum_{l_1}^{l_c} v_i[(a_ir_j + b_ir_k)P_\mathrm{w} + (a_ic_j + b_ic_k)xP_0 + (a_im_j + b_im_k)xP_1], P\right) \\ =& e\left(\sum_{l_1}^{l_c} v_i[(a_ir_j + b_ir_k)P_\mathrm{w} + c_ixP_0 + m_i'xP_1], P\right) \\ =& e\left(\sum_{l_1}^{l_c} v_i(a_ir_j + b_ir_k)P_\mathrm{w}, P\right)e\left(x\sum_{l_1}^{l_c} v_i(c_iP_0 + m_i'P_1), P\right) \\ =& e\left(\sum_{l_1}^{l_c} v_i(a_ir_j + b_ir_k)P, P_\mathrm{w}\right)e\left(\sum_{l_1}^{l_c}(v_ic_iP_0 + v_im_i'P_1), Q\right) \\ =& e\left(\sum_{l_1}^{l_c} v_iT_i', P_\mathrm{w}\right)e\left(\sum_{l_1}^{l_c} v_ic_iP_0 + \sum_{l_1}^{l_c} v_im_i'P_1, Q\right) = e(T_n', P_\mathrm{w})e\left(\sum_{l_1}^{l_c} c_iv_iP_0 + \mu'P_1, Q\right). \end{aligned} \tag{4}$$

temporarily to meet the auditing requirements by following the above steps.

Next, for simplicity, we will present a specific example of a signature forgery attack against Tan and Jia (2014)'s protocol with some artificially small parameters. We use the JPBC Library (http://gas.dia.unisa.it/projects/jpbc/) with 'type A pairing' to perform the forgery attack.

Assume that the auditing challenge is chal $= \{(1,1),(2,1),(3,3)\}$, and the block m_3 is corrupted or deleted for some reasons. Let $\{S_i, T_i\}$ represent the signature of block m_i. Let $\mathscr{A}_1$ be a malicious cloud server. As the integrity of the challenged data is broken, $\mathscr{A}_1$ has to perform the signature forgery attack in the following steps:

1. $\mathscr{A}_1$ extracts the values of $\{m_1 = 251, S_1 = (x_{s1}, y_{s1}), T_1 = (x_{t1}, y_{t1})\}$, $\{m_2 = 253, S_2 = (x_{s2}, y_{s2})$, and $T_2 = (x_{t2}, y_{t2}\}$, since the two blocks and corresponding signatures are stored correctly.

2. According to $c_i = H_3(\mathrm{ID}, \mathrm{filename}, \mathrm{id}_i)$, $\mathscr{A}_1$ computes c_1, c_2, c_3 and stores them.

3. $\mathscr{A}_1$ picks a random data block $m_3' = 255$ to replace block m_3.

4. By referring to Eq. (3), $\mathscr{A}_1$ first calculates the values of a and b, and then derives the signature $S_3^{'} = aS_0 + bS_1$ and $T_3^{'} = aT_0 + bT_1$.

5. $\mathscr{A}_1$ computes μ', S_n', and T_n'. It then sends $\{\mu', S_n', T_n'\}$ to the TPA.

6. The TPA verifies the received proof by

$$e(S_n', P) = e(T_n', P_{\mathrm{w}})e\left(\sum_1^3 c_i v_i P_0 + \mu P_1, Q\right). \quad (5)$$

Our experiment shows the above equation holds.

Due to the length of large integers, we save the value of the above parameters in Table 1, which can be used to verify the accuracy of this attack, and all these data are in hexadecimal format.

3.2 Data privacy attack

Tan and Jia (2014) claimed that the NaEPASC protocol is efficient and secure when verifying the integrity of outsourced data. In this subsection, we prove that an external attacker can extract the outsourced data by acting as a TPA, and then we give a toy example of the data recovery attack to further illustrate the attack.

Let $\mathscr{A}_2$ be an external attacker who can perform the auditing task by impersonating a public auditor. Suppose a data file M is composed of n blocks, i.e., $M = m_1 \parallel m_2 \parallel \cdots \parallel m_n$. In addition, $\mathscr{A}_2$ wants to disclose the block $m_{s_1}, m_{s_2}, \cdots, m_{s_c}$.

To obtain the content of the above data blocks, $\mathscr{A}_2$ will implement a data privacy attack after the setup phase through the following steps.

1. $\mathscr{A}_2$ selects a c-element subset $\{s_1, s_2, \cdots, s_c\}$ from the set $\{1, 2, \cdots, n\}$, which denotes the indices of challenged data blocks.

2. $\mathscr{A}_2$ queries the cloud server for at least c times during the challenge phase, and the auditing challenges are defined as

$$\begin{cases} \mathrm{chal}_1 = \{(s_1, v_{1s_1}), (s_2, v_{1s_2}), \cdots, (s_c, s_{1s_c})\}, \\ \mathrm{chal}_2 = \{(s_1, v_{2s_1}), (s_2, v_{2s_2}), \cdots, (s_c, s_{2s_c})\}, \\ \quad\vdots \\ \mathrm{chal}_c = \{(s_1, v_{cs_1}), (s_2, v_{cs_2}), \cdots, (s_c, s_{cs_c})\}. \end{cases} \quad (6)$$

3. In return, the cloud server responses to $\mathscr{A}_2$ with c corresponding auditing proofs as

$$\begin{cases} \mathrm{pf}_1 = \{\mu_1, S_{1n}, T_{1n}\}, \\ \mathrm{pf}_2 = \{\mu_2, S_{2n}, T_{2n}\}, \\ \quad\vdots \\ \mathrm{pf}_c = \{\mu_c, S_{cn}, T_{cn}\}, \end{cases} \quad (7)$$

where $\mu_i = \sum_{s_1}^{s_c} v_j m_j$, $S_{in} = \sum_{s_1}^{s_c} v_j S_j$, $T_{in} = \sum_{s_1}^{s_c} v_j T_j$, and $1 \le i \le c$.

4. $\mathscr{A}_2$ first collects some one-dimensional matrices from its challenges such as $\boldsymbol{v}_1 = [v_{1s_1}, v_{1s_2}, \cdots, v_{1s_c}]$, $\boldsymbol{v}_2 = [v_{2s_1}, v_{2s_2}, \cdots, v_{2s_c}]$, $\cdots$, $\boldsymbol{v}_c = [v_{cs_1}, v_{cs_2}, \cdots, v_{cs_c}]$. Then it sums up these matrices into the following matrix:

$$\boldsymbol{V} = \begin{bmatrix} v_{1s_1} & v_{1s_2} & \dots & v_{1s_c} \\ v_{2s_1} & v_{2s_2} & \dots & v_{2s_c} \\ \vdots & \vdots & & \vdots \\ v_{cs_1} & v_{cs_2} & \dots & v_{cs_c} \end{bmatrix}. \quad (8)$$

5. Let $\det(\boldsymbol{V}) \neq 0$. Then $\mathscr{A}_2$ computes a matrix $\boldsymbol{Y}$ which satisfies $\boldsymbol{YV} = \boldsymbol{E}$, where $\boldsymbol{E}$ is a unit matrix.

6. Let matrix $\boldsymbol{F} = [m_{s_1}, m_{s_2}, \cdots, m_{s_c}]^{\mathrm{T}}$, and $\mathscr{A}_2$ constructs a matrix $\boldsymbol{U} = [\mu_1, \mu_2, \ldots, \mu_c]^{\mathrm{T}}$, which is constructed by part of the auditing proofs from the cloud server.

7. As $\boldsymbol{U} = \boldsymbol{VF}$, $\mathscr{A}_2$ can succeed in obtaining the content of the data blocks from the equation $\boldsymbol{F} = \boldsymbol{YU}$.

In particular, $\mathscr{A}_2$ can go further and select other data block sets to challenge, so it can recover all the

Table 1 Parameters of the signature forgery attack

Parameter	Value(s)							
x	33e2a151	dec19b64	9b71e02e	26d642fe	4d0a0b88			
P	13f17424	63b556f3	b2b19e6c	6c368bc6	e7d2776e	9620d30e	aa79bd93	b5d7d68d
	861a575b	73e056fc	a34c616d	0e22ba54	8da46c38	35efc269	ab8c8b40	955d15cc
	1027db59	20039be2	8c042ab3	b7d4019f	ca02a9fd	dbd7dd4e	efef8d3e	ffb4845b
	443389f9	be22b73b	73bcb18b	28566fd0	1246ac69	9a3dc206	7dddd37b	fa6fd5a7
Q	95a7e384	59d4d37d	f41a6e01	05ffe991	1d8feb16	9e78ec91	55b2a7f8	9dd22bd3
	e0a58e6f	606f7105	262846aa	97b3de67	ee7d1ee6	80e46a14	62dde0ab	3ebe7676
	4f247377	570d61f5	80e7bf76	7d81c055	01ee797f	f59a23f7	d1db2d30	74ef0c1d
	22f38194	14d2cade	a314a40f	98b70789	31ff7fc2	29676a46	c5fda836	17c6b1c6
P_0	6a55b358	25d566b8	04ea1d8e	4b2b20b9	29d707ae	2f0c242f	f8629168	bccc6fd5
	c7e5d967	ecc53021	ed38efae	7e12036d	c0076d86	97320405	cd486dae	8def465a
	6fe605f6	9c20f7f0	57572610	0e0afd29	722c50a5	a5b8d4d4	c6b6d9af	b6f79043
	c680b478	b8a14798	3511bb42	06c2aab0	514316e7	d342ea89	f9c42603	b7fdc533
P_1	93ece5bb	198903c6	24590770	2f13030d	39d7de9f	a33b9ed1	403a0ab7	1ef5c784
	3d1a8c27	79f4f202	b70b4631	a18373b5	c2eaccac	a576235a	6a2897b5	fc356904
	5b95ce5a	2f7a5a61	ef570e23	5015a5de	dc183d19	f6de8bba	fbc2eab7	92de5bfd
	407930ab	36ae4619	66a9d9c1	b2858a20	47cb08ba	adae27d7	1b3cdc84	5bf97620
P_{w}	1885ab50	ca047db6	84d164e3	46eb64bb	50477910	853204df	117151f9	46386715
	792720af	bb0d8d06	d51724eb	b7e4526d	592190d9	b5294a7b	edbddce7	26b52694
	27406cb2	9f67dcef	60a3badf	c02efbdf	9aa9cd17	714374e1	0367c460	7a3ca5d1
	c82c5fad	2f4b9609	941b646e	f5ceb93a	354f2d15	e7d5a788	80200b45	4870fe3e
r_1	27727ccf	025f9f79	b4d9fbaf	efe9e2b4	b473ecf7			
r_2	1b3e5867	d96accaf	c6c37273	0e6f942a	c1604ace			
S_1	83c88c4b	85d564c2	6d3fdf51	aecfa114	c1d2ee01	ca2c8b63	1a2b46f9	f71ee98c
	47f2ecd2	d0497d79	4c2808b5	ee537eb7	55d146b3	96d74e05	b8af5c13	d59891fe
	21a815fd	b976822e	5feb6023	b117315d	e48fac54	332ae067	cd4b6ddf	ffcf8fa0
	bf9c4479	9f3df154	3dde4382	0aa4a3e6	e990bcdd	a31949ac	edc03473	2042b43c
S_2	1d0626ae	b2fe1081	6b9c277d	ffc167c2	43745d5e	c285efdd	b53d9df9	6b5eacef
	992477a8	afece68d	ca0a3251	3cb35b35	5fb2819f	0b82dc0f	0e111163	a622b42d
	17097a86	8bec4e6d	539b9ef2	29e05e16	090023e8	e09d5b2f	ae9f8e9a	199ab4a3
	d073af6e	22e12dda	1b8d1fde	b8626820	9fb6c3b9	8eb1c45a	eeda68a8	a252d4be
a	80000000	00000800	00000000	00000000	00000000			
b	00000002							
$S_3^{'}$	901114cb	2e5a92a2	1eee1dd0	16acc6d1	04bdab93	9dfb7f82	0270d923	763a0360
	66120aa7	d3205d22	6d65add1	5c821c33	e3db9b5a	8036f447	3c2578a4	21f6551f
	8cbbcb71	cd09bdf0	affd30bc	427c442d	38fe7297	c798dc38	dad51817	0947a86d
	2b5cba05	9288d992	5eeb5c1d	56ea0186	61f59847	0d892f0a	b0b2e2b6	b48ca72a
$T_3^{'}$	149f7356	a859554f	5a16a040	bbc35a29	02cbeb95	7897287a	0ab2b468	7d842a20
	49b22695	703d8236	36dd52ca	9764cff9	433512e6	b43cc8fd	1311789d	269d206d
	7465267d	a98fe305	cced1495	7860f185	02c2e64b	d46a5032	0f7e6c0a	7f8d4011
	f344fd2c	8b250b8c	20ef0a11	fe97884d	70f421cb	4b8e359b	aa0adf0b	e1e6d7b6

block data of the data file. Therefore, NaEPASC cannot preserve the privacy of outsourced data in the cloud.

Now, we present a simple example to prove that the curious TPA can actually reveal the content of the challenged data blocks. For simplicity, we select some parameters that are as small as possible:

1. The adversary $\mathscr{A}_2$ acts as a TPA to perform the auditing challenges. Now, it first selects a three-element subset $I = \{1, 2, 3\}$. It first selects three random values $\{v_{11} = 1, v_{12} = 1, v_{13} = 4\} \in Z_p$, and then sends $\text{chal}_1 = \{(1, 1), (2, 1), (3, 4)\}$ as the auditing challenge to the cloud server.

2. Upon receiving chal_1, the cloud server computes

$$\begin{cases} \mu_1 = \sum_{i=1}^{3} v_{1i} m_i = 1 \times 3586 + \\ \qquad 1 \times 3587 + 4 \times 3588 = 21\ 525, \\ S_{1n} = \sum_{i=1}^{3} v_{1i} S_i, T_{1n} = \sum_{i=1}^{3} v_{1i} T_i, \end{cases} \tag{9}$$

where $m_1 = 3586$, $m_2 = 3587$, and $m_3 = 3588$.

3. Next, $\mathscr{A}_2$ repeats step 1 twice. It always chooses the same data blocks $\{m_1, m_2, m_3\}$

as the challenged blocks but with different random values v_{ji}. For example, let $\{v_{21} = 1, v_{22} = 2, v_{23} = 3\}$ and $\{v_{31} = 1, v_{32} = 5, v_{33} = 5\}$. Then, it sends $\text{chal}_2 = \{(1,1),(2,2),(3,3)\}$ and $\text{chal}_3 = \{(1,1),(2,5),(3,1)\}$ to the cloud server.

4. Similar to step 2, the cloud server returns the auditing proofs $\{\mu_2, S_{2n}, T_{2n}\}$ and $\{\mu_3, S_{3n}, T_{3n}\}$ to $\mathscr{A}_2$ after receiving the auditing challenge chal_2 and chal_3, where $\mu_2 = \sum_{i=1}^{3} v_{2i}m_i = 1\times 3586 + 2\times 3587 + 3\times 3588 = 21\,524$, $\mu_3 = \sum_{i=1}^{3} v_{3i}m_i = 1\times 3586 + 5\times 3587 + 1\times 3588 = 25\,109$.

5. Now, $\mathscr{A}_2$ can obtain the matrix $\boldsymbol{V}$, constructed by a random value v_{ji}, and another matrix $\boldsymbol{Y}$ derived from the formula $\boldsymbol{YV} = \boldsymbol{E}$, as follows:

$$\boldsymbol{V} = \begin{bmatrix} 1 & 1 & 4 \\ 1 & 2 & 3 \\ 1 & 5 & 1 \end{bmatrix}, \boldsymbol{Y} = \begin{bmatrix} -13 & 19 & -5 \\ 2 & -3 & 1 \\ 3 & -4 & 1 \end{bmatrix}.$$

6. Let $\boldsymbol{U} = [\mu_1, \mu_2, \mu_3]^{\mathrm{T}} = [21525, 21524, 25109]^{\mathrm{T}}$. $\mathscr{A}_2$ computes the value of $\boldsymbol{YU}$ and the result is

$$\begin{bmatrix} -13 & 19 & -5 \\ 2 & -3 & 1 \\ 3 & -4 & 1 \end{bmatrix} \begin{bmatrix} 21\,525 \\ 21\,524 \\ 25\,109 \end{bmatrix} = \begin{bmatrix} 3586 \\ 3587 \\ 3588 \end{bmatrix}.$$

Because the data blocks sampled for checking are exactly $m_1 = 3586$, $m_2 = 3587$, and $m_3 = 3588$, $\mathscr{A}_2$ succeeds in disclosing the content of the challenged data blocks.

4 Conclusions

In this paper, we first reviewed the NaEPASC protocol proposed by Tan and Jia (2014), and then pinpointed the insecurity in both the setup phase and the challenge phase. The cryptanalysis demonstrates that a malicious cloud server can impersonate the cloud user to generate a valid signature so that it can pass the verification of TPA without correct data storage. Meanwhile, the analysis shows that NaEPASC cannot maintain the privacy of the data, because the TPA can reveal the content of outsourced data through auditing challenges and proofs. Hence, NaEPASC is not suitable for checking the storage correctness of outsourced data as a public auditing protocol. Although we have not conceived a good idea for solving the problems, our work can help cryptographers and engineers design and implement more secure and efficient identity-based public auditing schemes for cloud storage.

References

Ateniese, G., Burns, R., Curtmola, R., *et al.*, 2007. Provable data possession at untrusted stores. Proc. 14th ACM Conf. on Computer and Communications Security, p.598-609. https://doi.org/10.1145/1315245.1315318

Chen, B., Curtmola, R., 2012. Robust dynamic provable data possession. 32nd Int. Conf. on Distributed Computing Systems Workshops, p.515-525. https://doi.org/10.1109/ICDCSW.2012.57

Fu, Z.J., Sun, X.M., Liu, Q., *et al.*, 2015. Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Trans. Commun.*, **E98.B**(1):190-200. https://doi.org/10.1587/transcom.E98.B.190

Fu, Z.J., Ren, K., Shu, J.G., *et al.*, 2016. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Trans. Parall. Distrib. Syst.*, **27**(9):2546-2559. https://doi.org/10.1109/TPDS.2015.2506573

Guo, P., Wang, J., Geng, X.H., *et al.*, 2014. A variable threshold-value authentication architecture for wireless mesh networks. *J. Intern. Technol.*, **15**(6):929-935. https://doi.org/10.6138/JIT.2014.15.6.05

He, D.B., Zeadally, S., Wu, L.B., 2015. Certificateless public auditing scheme for cloud-assisted wireless body area networks. *IEEE Syst. J.*, in press. https://doi.org/10.1109/JSYST.2015.2428620

Li, J.T., Zhang, L., Liu, J.K., *et al.*, 2016. Privacy-preserving public auditing protocol for low performance end devices in cloud. *IEEE Trans. Inform. Forens. Secur.*, **11**(11): 2572-2583. https://doi.org/10.1109/TIFS.2016.2587242

Liu, J.K., Au, M.H., Huang, X., *et al.*, 2016. Fine-grained two-factor access control for web-based cloud computing services. *IEEE Trans. Inform. Forens. Secur.*, **11**(3): 484-497. https://doi.org/10.1109/TIFS.2015.2493983

Ren, Y.J., Shen, J., Wang, J., *et al.*, 2015. Mutual verifiable provable data auditing in public cloud storage. *J. Intern. Technol.*, **16**(2):317-323. https://doi.org/10.6138/JIT.2015.16.2.20140918

Shacham, H., Waters, B., 2008. Compact proofs of retrievability. *LNCS*, **5350**:90-107. https://doi.org/10.1007/978-3-540-89255-7_7

Shacham, H., Waters, B., 2013. Compact proofs of retrievability. *J. Cryptol.*, **26**(3):442-483. https://doi.org/10.1007/s00145-012-9129-2

Tan, S., Jia, Y., 2014. NaEPASC: a novel and efficient public auditing scheme for cloud data. *J. Zhejiang Univ.-Sci. C (Comput. & Electron.)*, **15**(9):794-804. https://doi.org/10.1631/jzus.C1400045

Wang, C., Chow, S.S.M., Wang, Q., *et al.*, 2013. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.*, **62**(2):362-375. https://doi.org/10.1109/TC.2011.245

Xia, Z., Wang, X., Sun, X., *et al.*, 2016. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Trans. Parall. Distrib. Syst.*, **27**(2):340-352. https://doi.org/10.1109/TPDS.2015.2401003