JOURNAL OF INFORMATION SECURITY AND APPLICATIONS 23 (2015) 68–73

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jisa

Cryptanalysis of an integrity checking scheme for cloud data sharing ☆

Yuan Zhang *, *Chunxiang Xu*, *Jining Zhao*, *Xiaojun Zhang*, *Junwei Wen*

School of Computer Science and Engineering, University of Electronic Science and Technology of China, 2006 Xi Yuan Avenue, West High-tech Zone, Chengdu 611731, China

ARTICLE INFO

Article history:
Available online 12 June 2015

Keywords:
Integrity checking
Cloud data sharing
Cryptanalysis

ABSTRACT

Cloud storage provides an efficient way for users to work together as a group by sharing data with each other. However, since shared data can be accessed and modified by multiple users and group membership may be changed frequently, this new paradigm poses many challenges for keeping integrity of shared data. Recently, Yuan et al. proposed an efficient integrity checking scheme (IEEE INFOCOM 2014, doi: 10.1109/INFOCOM.2014.6848154) for cloud data sharing with multi-user modification, which had many appealing features. They claimed that the scheme is secure and efficient, and they also provided the formal security proof and the performance evaluation. Regretfully, existing two security flaws in Yuan et al.'s scheme are pointed out in this letter. Specifically, by fooling the third-party auditor (TPA) into trusting that the data is well maintained by the cloud server, an adversary can process the following two deceiving methods. Firstly, the adversary can modify the shared data and tamper with the interaction messages between the cloud server and the TPA, thus invalidating shared data integrity checking. Secondly, an adversary, who records a fraction of the cloud-stored data, can overwrite the vast majority of the shared data by using the recorded data and passing shared data integrity verification. Furthermore, we suggest a solution to the two security flaws while retaining all the desirable features of the original scheme.

1. Introduction

Cloud storage is a momentous service of cloud computing, which provides users an easy, cost-effective and reliable way to manage data. Using cloud storage services, users can access their data remotely through the internet without incurring the substantial hardware, software and personnel costs involved in deploying and maintaining applications in local storage. However, a practical cloud environment is so complex that there are some external and internal threats in it (Lee et al., 2013; Li et al., 2009). Typically, the cloud-stored data may be corrupted in any infrastructure, no matter what high degree of reliable measures the cloud server would provide (Yang and Jia, 2013). In addition, an external adversary with profits motivation may tamper with the cloud-stored data but convinces the cloud user of the data correctness and integrity (Dolev and Yao, March, 1983; Ni et al. ,2013; Xu et al., 2014). Therefore, due to the existence of external and internal threats to the practical cloud environment, the integrity of the outsourced data has naturally become the concerned focus of cloud users (Yang and Jia 2012).

In order to ensure the integrity of data stored on the cloud, several schemes have been proposed based on various

☆ This article belongs to the special issue Security, Privacy and Trust in Future Networks and Mobile Computing.

* *Corresponding author.*
E-mail addresses: ZY_LoYe@126.com, ZY_LoYe@qq.com (Y. Zhang).

http://dx.doi.org/10.1016/j.jisa.2015.05.001

techniques (Wang et al., 2010; Wang et al., 2013a; Worku et al., 2014; Zhao et al., December 2013; Wang et al., 2012; Wang et al., 2013b; Wang et al., 2015). In these schemes, a cloud user resorts to a professional and capable third-party auditor (TPA) to audit the outsourced data when needed. This provides a feasible and efficient way for the cloud users to ensure the integrity of their data in the cloud. However, most of these schemes only support the case of single writers, that is, only the data owner holds the secret keys and modifies his data that is outsourced into the cloud server. If these schemes are simply extended to support multiple writers with data integrity checking, the data owner will face a tremendous workload, since he has to stay online, receive the modified data from other users and generate the new authentication tags for them (Yuan and Yu, 2014). Therefore, there are many barriers to applying these schemes to different practical scenarios.

To support cloud-stored data that can be modified by multiple users, recently, Yuan et al. proposed an efficient public integrity checking scheme for cloud data sharing with multi-user modification. In the scheme, the data owner, who is named as the master user, outsources his data into the cloud server. He designates some cloud users as a group, shares the data with all the group users and removes any valid group users if needed. The shared data can be modified by any valid group user and the user who last modifies the shared data can be identified. Moreover, collusion among the cloud server and any group user other than the master user will not give users any chance to impersonate other valid users. The scheme also can be extended to support batch verification to improve efficiency. The authors claimed that their scheme is secure and the proof information in the scheme cannot be forged, they also gave formal security proof to buttress their views.

In this letter, we first review the Yuan et al.'s scheme and point out that the scheme has two security flaws. Accordingly, two deceiving methods can work on the scheme and invalidate the shared data integrity checking. Particularly, we first show that an adversary modifies the shared data to his needs and records how the shared data is modified, according to the interaction messages between the cloud server and the TPA, he can generate valid proof information to deceive the TPA and pass the shared data integrity checking. We also show that if an adversary can intrude on the cloud server and record a fraction of the shared data, with the second security flaw, he can overwrite the vast majority of the shared data by using the recorded data. In this way, even if the shared data has been corrupted, it still can pass the TPA's shared data integrity checking. Finally, for each of the two security flaws, we suggest a solution to fix it without sacrificing any desirable features of the original scheme.

2. Review the Yuan et al.'s scheme

In the scheme, there are K users in a group sharing data stored on the cloud server. The manager of the group is the master user u_0, he is also the data owner. That is, u_0 uploads his data to the cloud server and manages membership of the group. All users in the group can access and modify the data. The data file F is divided into n blocks and each block is consisted of s elements. The data component is denoted as $F=\{m_{i,j}\}_{i\in[1,n],j\in[0,s-1]}$. Let $H(\cdot)$ denotes the one-way hash function, G and G_1 be two multiplicative cycle groups of prime order q, g and u be two random generators of G. $e{:}G\times G\to G_1$ is a bilinear pairing. $f_{\vec{a}}(x)$ denotes a polynomial with coefficient vector $\vec{a}=(a_0,a_1,\ldots,a_{s-1})$. Here, For ease of description, we omit the batch auditing and any other inessential details. The basic scheme involves seven algorithms: **KeyGen**, **Setup**, **Update**, **Challenge**, **Prove**, **Verify** and **User Revocation**. Because of limited space we will not review the **User Revocation** algorithm in this Section, see (Yuan and Yu, 2014) for more details.

2.1. KeyGen

Firstly, each user u_k, $0\le k\le K-1$ in the group randomly chooses $\varepsilon_k\leftarrow Z_q^*$ and computes $\kappa_k=g^{\varepsilon_k}$. For group users u_k, $1\le k\le K-1$, they compute $g^{\frac{1}{\varepsilon_k}}$ and send it to the master user u_0. The master user computes $v=g^{\alpha\varepsilon_0}$ and $g^{\frac{\varepsilon_0}{\varepsilon_k}}$ for $1\le k\le K-1$. u_0 randomly chooses $\alpha\leftarrow Z_q^*$ and computes g^{α^j}, $0\le j\le s+1$. The public keys of the system $PK=\left\{g,u,q,\left\{g^{\alpha^j}\right\}_{0\le j\le s+1},\left\{\kappa_k,g^{\frac{\varepsilon_0}{\varepsilon_k}}\right\}_{0\le k\le K-1}\right\}$, the mast keys of the system $MK=\{\in_0,\alpha\}$ and the secret keys of users $SK_k=\{\varepsilon_k\}_{1\le k\le K-1}$.

2.2. Setup

The master user u_0 splits the data file F into n data blocks, and further splits every block into s elements. Then, u_0 computes authentication tag σ_i for each data block as:

$$\sigma_i=\left(u^{B_i}\cdot\prod_{j\in[0,s-1]}g^{m_{i,j}\alpha^{j+2}}\right)^{\varepsilon_0}=\left(u^{B_i}\cdot g^{f_{\vec{m}_i}(\alpha)}\right)^{\varepsilon_0}$$

where $B_i=H(\{z_i\|H(m_i)\|0\})$, z_i is the index of m_i and $\vec{m}_i=\{m_{i,0},m_{i,1},\ldots,m_{i,s-1}\}$. u_0 uploads data blocks and the corresponding authentication tags σ_i to the cloud server, and sends B_i to the TPA (in practical implementation B_i can be first uploaded to cloud and later on downloaded by TPA).

2.3. Update

When a group user u_k $(k\ne 0)$ modifies a data block m_i to m_i'. He computes the tag of the new data block as:

$$\sigma_i'=\left(u^{B_i}\cdot\prod_{j\in[0,s-1]}g^{m_{i,j}'\alpha^{j+2}}\right)^{\varepsilon_k}$$

where $B_i=H(\{z_i\|H(m_i')\|k\})$. Then u_k uploads m_i' and σ_i' to the cloud server, and sends B_i to the TPA (in practical implementation B_i can be first uploaded to cloud and later on downloaded by TPA).

2.4. Challenge

To verify the data integrity, the TPA first randomly chooses d data blocks, these data blocks' indices compose a set D. Suppose the chosen d data blocks are collectively modified by a set of