Expert Systems with Applications 41 (2014) 7789–7796

Contents lists available at ScienceDirect

Expert Systems with Applications

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

Improved security of a dynamic remote data possession checking protocol for cloud storage

Yong Yu [a,b], Jianbing Ni [a,*], Man Ho Au [c], Hongyu Liu [a], Hua Wang [a], Chunxiang Xu [a]

[a] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[b] State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[c] Department of Computing, The Hong Kong Polytechnic University, Hong Kong

ARTICLE INFO

Article history:
Available online 19 June 2014

Keywords:
Cloud storage
Data possession checking
Homomorphic hashing
Dynamic auditing

ABSTRACT

Cloud storage offers the users with high quality and on-demand data storage services and frees them from the burden of maintenance. However, the cloud servers are not fully trusted. Whether the data stored on cloud are intact or not becomes a major concern of the users. Recently, Chen et al. proposed a remote data possession checking protocol to address this issue. One distinctive feature of their protocol support data dynamics, meaning that users are allowed to modify, insert and delete their outsourced data without the need to re-run the whole protocol. Unfortunately, in this paper, we find that this protocol fails to achieve its purpose since it is vulnerable to forgery attack and replace attack launched by a malicious server. Specifically, we show how a malicious cloud server can deceive the user to believe that the entire file is well-maintained by using the meta-data related to the file alone, or with only part of the file and its meta-data. Then, we propose an improved protocol to fix the security flaws and formally proved that our proposal is secure under a well-known security model. In addition, our improvement keeps all the desirable features of the original protocol.

1. Introduction

Cloud storage provides a novel service model (Wu, 2011) in which data are maintained, managed and backed up remotely and accessed by cloud users over the network at anytime and from anywhere (Jula, Sundararajan, & Othman, 2014). Nowadays, an increasing number of organizations and individuals would like to outsource their data to cloud to enjoy appealing advantages of cloud storage. However, once a data owner uploads his/her data to cloud and delete the local copy of the files, the owner loses physical control over the outsourced data.

Naturally, integrity and confidentiality of the data are of prime concern in this scenario. Indeed, in the white paper entitled "Cloud Computing Vulnerability Incidents: A Statistical Overview"[1] by the Cloud Vulnerabilities Working Group of the cloud security alliance (CSA), Data Loss & Leakage is the second most frequent incident types in the seven threat types defined by CSA. Some examples are from the prominent providers (e.g. Amazon,[2] Evernote[3]). The same white paper stated that "...the data collected is the result of a best effort attempt." since it is not mandatory for the providers to report these incidents. On the contrary, the cloud providers are not fully trusted (Wang, Zeng, & Yao, 2012) and it might be of their interest to hide data loss incidents in order to maintain their reputation.

To improve accountability of the cloud server, it is therefore desirable to have the cloud server provide evidence to convince its users that their data are not tempered with nor discarded periodically (Lin & Chang, 2011). The major research problem in this setting is that the users do not have a local copy of the data, meaning that traditional integrity mechanism (e.g. digital signature) is not suitable as it requires the user to download the data from the cloud, which is costly or sometimes infeasible.

To check the integrity of remote data, in 2007, Ateniese et al. (2007, 2011) presented the notion of provable data possession (PDP) and constructed two efficient and provably secure PDP schemes based on homomorphic verifiable tags. In their protocols, cloud users are allowed to verify data integrity (Kamel, 1995) without retrieving the entire file. At the same time, Juels, Burton, and Kaliski (2007) defined the model of proof of retrievability (PoR) which allows the server to construct a concise proof to convice the cloud user that their data can be retrieved, and proposed a sentinel-based PoR construction utilizing error-correcting code. In 2008, Shacham and Waters (2008, 2013) described two efficient

* Corresponding author. Tel.: +86 15882402247.
E-mail address: nimengze@gmail.com (J. Ni).

[1] https://cloudsecurityalliance.org/research/vulnerabilities/#_downloads.
[2] http://www.businessinsider.com.au/amazon-lost-data-2011-4.
[3] http://cloutage.org/incidents/125-evernote-corporation-evernote.

http://dx.doi.org/10.1016/j.eswa.2014.06.027
0957-4174/

and compact PoR schemes. The first one is a public verifiable PoR scheme built from the signature algorithm due to Boneh, Lynn and Shacham (referred to as BLS signature hereafter) (Boneh, Lynn, & Shacham, 2001), and the other one is a private verifiable PoR scheme based on the pseudo-random function. In 2009, Ateniese, Kamara, and Katz (2009) put forward a framework for building publicly-verifiable PDP scheme with an unbounded number of verifications from public-key homomorphic linear authenticator which can be generated from any identification protocol. In the following, we use the term remote data possession checking (RDPC) protocol to refer to any protocol (including PDP and PoR) that aims to solve the integrity of remote data.

With the proliferation of cloud storage, a number of data auditing protocols such as Chen (2013), Wang, Ren, Lou, and Li (2010, 2013) and Zhu, Hu, Ahn, and Stephen (2012a, 2012b, 2012) were proposed to ensure the integrity of the outsourced data. The aforementioned research focus on static data in which the outsourced data are not going to be modified. Recently, several PDP or PoR schemes (Ateniese, Pietro, Mancini, & Tsudik, 2008; Erway, Kupcu, Papamanthou, & Tamassia, 2009; Wang, Wang, Ren, Lou, & Li, 2009, 2012; Yang & Jia, 2013) supporting dynamic data operations were proposed as well. In particular, a recent scheme by Chen, Zhou, Huang, and Xu (2013) supports the most general forms of data operation, such as block modification, insertion and deletion. This protocol is based on a homomorphic hash algorithm, in which the hash value of the sum of two blocks is equal to the product of two hash values. To support data updating, the Merkle Hash Tree (MHT) is employed to record the location for each data operation. Chen et al. also demonstrated their proposal compares favourably with the state-of-the-art protocols and they concluded that the performance is limited by network bandwidth rather than cryptographic operations.

Our contribution. The contributions of this paper are threefold.

(1) We identify several security flaws in the dynamic RDPC protocol in Chen et al. (2013). As long as the authenticated data structure, Merkle Hash Tree, is well maintained, the server can always generate a valid proof by using forgery attack or replace attack to cheat the user that the data are well accommodated in cloud, while actually some data blocks may have been corrupted. This means the protocol cannot achieve its design goals and cannot be adopted in real-world applications.
(2) We propose an improved dynamic RDPC protocol to mend these security weaknesses by making use of some techniques including modifying the homomorphic hash functions to be cryptographically secure, involving the hash value of the data block in generating each tag, and the random sampling trick to improve the efficiency of the protocol.
(3) We prove the security of the fixed protocol in the well-known security model due to Ateniese et al. (2007), and show the improvement maintains all the desirable features of the original protocol.

Organization: The rest of the paper is organized in the following way. Section 2 gives some preliminaries used in this paper. Section 3 reviews the dynamic RDPC protocol in Chen et al. (2013) and presents our security analysis on the protocol. Section 4 comes up with our improved protocol and its performance. Section 5 describes the security proof of the new protocol, and Section 6 concludes the paper.

2. Preliminaries

In this section, we review some preliminary knowledge used in this paper, including the homomorphic hash functions and Merkle Hash Tree.

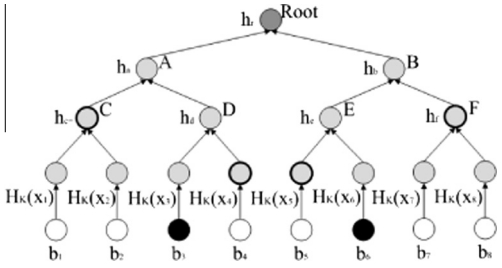

Fig. 1. Merkle Hash Tree.

2.1. Homomorphic hash functions

A homomorphic hash function $H(\cdot)$ (Krohn, Freeman, & Mazieres, 2004) consists of two sub-algorithms, namely, homomorphic key generation and hash generation. In the first phase, it takes as input four security parameters $(\lambda_p, \lambda_q, m, s)$, where λ_p and λ_q are discrete log security parameters, m is the number of sectors in per hash message, s is a random seed which can be generated by hashing file name, and outputs the homomorphic key $K = (p, q, g)$, where p and q are random primes satisfying $|p| = \lambda_p$, $|q| = \lambda_q$ and $q|(p-1)$, $g = \{g_1, g_2, \ldots, g_m\}$ is $1 \times m$ row vector of order q in Z_p.

In hash generation, a message F is divided into n blocks, say, $F = b_1||b_2||\cdots||b_n$, and each block is further segmented into m sectors $b_i = b_{i1}||b_{i2}||\cdots||b_{im}$. The homomorphic hash value of F is $H_K(F) = (H_K(b_1), H_K(b_2), \ldots, H_K(b_n))$, and each $H_K(b_i)$ is computed as

$$H_K(b_i) = \prod_{j=1}^{m} g_j^{b_{ij}} \bmod p.$$

2.2. Merkle Hash Tree

A Merkle Hash Tree (MHT) (Merkle, 1980) is an authenticated data structure, which is used to efficiently and securely prove that a set of elements are undamaged and unaltered. It is constructed as a binary tree where leaves are the hash values $h(\cdot)$ of authentic data, in which $h(\cdot) : \{0,1\}^* \to \{0,1\}^c$ denotes a cryptographic hash function. Fig. 1 depicts an example of authentication. A verifier with the authentic h_r requests for (b_3, b_6) and requires the authentication of the received blocks. The prover provides the verifier with the auxiliary authentication information $\Omega_3 = \langle H_K(b_4),\ h_c\rangle$ and $\Omega_6 = \langle H_K(b_5),\ h_f\rangle$. Then the verifier can verify x_3 and x_6 by computing $h_d = h(H_K(b_3)||H_K(b_4))$, $h_e = h(H_K(b_5)||H_K(b_6))$, $h_a = h(h_c||h_d)$, $h_b = h(h_e||h_f)$ and $h_r = h(h_a||h_b)$, and then checking if the computed h_r is the same as the authentic one.

3. Security analysis of the dynamic RDPC protocols

In this section, we review the components, security requirements and the construction of the dynamic RDPC protocol in Chen et al. (2013) and show that it is vulnerable to forgery attack and replace attack.

3.1. Components of a dynamic RDPC protocol

A remote data possession checking protocol consists of the following algorithms (Ateniese et al., 2011): KeyGen, TagGen, Challenge, ProofGen, ProofVerify.

- KeyGen is a probabilistic algorithm run by a cloud user. It takes a security parameter k as input and returns K as the secret key of the user.