Journal of Universal Computer Science, vol. 21, no. 3 (2015), 473-482
submitted: 30/6/14, accepted: 28/2/15, appeared: 1/3/15

Insecurity of an Efficient Privacy-preserving Public Auditing Scheme for Cloud Data Storage

Hongyu Liu, **Leiting Chen**
(School of Computer Science and Engineering
University of Electronic Science and Technology of China
Chengdu, 610054, China
lhy@uestc.edu.cn, richardchen@uestc.edu.cn)

Zahra Davar, **Mohammad Ramezanian Pour**
(School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW 2522, Australia
zd991@uowmail.edu.au, mr417@uowmail.edu.au)

Abstract: Cloud storage has a long string of merits but at the same time, poses many challenges on data integrity and privacy. A cloud data auditing protocol, which enables a cloud server to prove the integrity of stored files to a verifier, is a powerful tool for secure cloud storage. Wang *et al.* proposed a privacy-preserving public auditing protocol, however, Worku *et al.* found the protocol is seriously insecure and proposed an improvement to remedy the weakness. In this paper, unfortunately, we demonstrate that the new protocol due to Worku *et al.* fails to achieve soundness and obtains merely limited privacy. Specifically, we show even deleting all the files of a data owner, a malicious cloud server is able to generate a response to a challenge without being caught by TPA in their enhanced but unrealistic security model. Worse still, the protocol is insecure even in a correct security model. For privacy, a dishonest verifier can tell which file is stored on the cloud. Solutions to efficient public auditing mechanisms with perfect privacy protection are still worth exploring.

Key Words: Cloud storage, Data integrity, Privacy-preserving, Security analysis
Category: H.2, H.3.7, H.5.4

1 Introduction

Cloud storage is becoming increasingly popular because of a laundry list of advantages of this kind of novel storage model. Currently, many cloud storage services such as Amazon S3, Google Cloud, and Microsoft Skydrive have attracted millions of users all over the world, including individuals and organizations. The flexibility and on demand manner of cloud storage brings a lot of appealing benefits over traditional storage approach, say, relief of the burden of storage management, avoiding capital expenditure on hardware, software and personnel maintenance, access to data with independent geographical locations [Armbrust et al. 2010]. More and more users would like to outsource their data to remote cloud servers seeking to reduce the maintenance and storage cost such that they can focus more on their core competencies.

Despite of a long string of merits, cloud storage does trigger many challenging security problems [Wei et al. 2014]. Since data owners lose the control over their data, a major concern of cloud users is whether their data keeps virgin since some important data as well as confidential files may be hosted on the cloud. Two main reasons may lead to the loss of data constantly [Wang et al. 2010, Wang et al. 2011]. Firstly, frequent data access increases the probability of disk corruption, but cloud servers would try to hide data loss incidents in order to maintain their reputations. Secondly, cloud servers are not necessarily fully trusted and consequently, malicious servers might discard the data that have not been or are rarely accessed for monetary reasons. As a result, a strong evidence that their data accommodated on cloud keeps unchanged and is not being tampered with or partially deleted is highly essential for cloud users.

However, traditional cryptographic primitives for data integrity checking such as hash functions and digital signatures cannot be applied to cloud storage scenario directly because a copy of original message is required in the verification of these technologies, while the data owner or a verifier does not keep such a copy in cloud environment. Retrieving the entire file from cloud for integrity verification is unpractical since the data stored on cloud is massive, or even big data. In 2007, Ateniese *et al.* proposed the notion of provable data possession(PDP) [Atenisese et al. 2007, Ateniese et al. 2011] for validating data integrity over remote servers to address this issue. In a typical PDP system, a data owner generates some metadata for a file, and then stores files together with the corresponding metadata to cloud. The data owner can check the integrity of stored data via a challenge-response protocol with the remote server. To generate a proof that the server hosts the file in its original form, the server computes a response using the data owner's challenge, the challenged data blocks and the metadata. The data owner validates the file is not being tampered with by checking the correctness of the response. At the same time, Juels *et al.* presented the concept of proof of retrievability (POR) [Jules et al. 2007], in which both error-correcting codes and spot-checking are employed to achieve the properties of integrity and retrievability of files. Subsequently, Shachem and Waters proposed compact proof of retrievability and constructed an elegant scheme from BLS short signature [Shacham et al. 2008, Shacham et al. 2012]. This construction [Shacham et al. 2008, Shacham et al. 2012] has been widely used as a building block to construct cloud data auditing protocols with additional features due to the beautiful properties of BLS signature scheme. Subsequently, PDP became a research hotspot and a variety of PDP schemes along with their analysis and improvement were proposed [Wang et al. 2010, Wang et al.2013, Yu, Niu et al. 2014, Yu, Ni et al. 2014]. Among which, public verifiability [Wang et al. 2010], batch auditing [Wang et al. 2010], and data privacy against verifiers [Wang et al.2013] are three advanced features of cloud data auditing for practical cloud storage purpose. In 2010, Wang *et al.* extended the protocol due to Shacham and Waters [Shacham et al. 2008] and proposed privacy-preserving public auditing scheme [Wang et al. 2010] for data storage security. However, Worku *et al.* [Worku et al. 2014] found that the scheme [Wang et al. 2010] is vulnerable to attacks from malicious cloud server and outside attackers regarding to storage correctness. To remedy the security weakness of this scheme, Worku *et al.* described an improvement in [Worku et al. 2014] which was claimed as being secure, with better efficiency and can support batch auditing. They also provided a comprehensive security analysis on storage correctness and privacy-preserving guarantee.

Contribution. In this paper, we show the construction in [Worku et al. 2014] is not secure in their security model or in a correct security model. To be specific, with the aid of signature queries, a malicious cloud server could generate a valid response to a challenge from a third party auditor (TPA) even the server has deleted all the files of a user or has corrupted the file. Regarding the data privacy, what the scheme can achieve is that an adversary cannot recover the entire file from the auditing process, which is similar to the one-wayness of encryption. We will show that it cannot achieve the IND-privacy introduced recently by Fan *et al.* [Fan et al. 2013].

2 Review of privacy-preserving public auditing scheme

In this section, we firstly review some preliminaries used in the paper and then recall the privacy-preserving public auditing scheme in [Worku et al. 2014].

Bilinear Map [Boneh et al. 2001]. G and G_T denote two multiplicative cyclic groups of the same prime order p. e denotes a bilinear map that for all $g, h \in G$ and $a, b \in Z_p^*$, $e(g^a, h^b) = e(g, h)^{ab}$. For such kind of bilinear map, there exists a computable algorithm that can compute e efficiently and $e(g, g) \neq 1$.

Notation. The data owner preprocesses the outsourced files by dividing each file F into n blocks $F = (m_1, m_2, \cdots, m_n)$ for $m_i \in Z_p (i = 1, \cdots, n)$. $H() : \{0,1\}^* \to G$ denotes a secure map-to-point hash function employed in BLS signature [Boneh et al. 2001] while $h() : G \to Z_p$ represents a secure hash function which maps elements of G uniformly to Z_p. $\pi_{key} : \{0,1\}^{log_2(n)} \times K \to \{0,1\}^{log_2(n)}$ and $f_{key} : \{0,1\}^* \times K \to Z_p$ denote a pseudorandom permutation and a pseudorandom function respectively, where key belongs to a key space K.

Scheme Review. The privacy-preserving public auditing scheme in [Worku et al. 2014] consists of the following algorithms.

KeyGen(1^k)**.** The data owner first generates a random signing key pair (ssk, spk), and then picks $x \in Z_p$, $u \in G$ and computes $v = g^x \in G$. The secret key is $sk = (x, ssk)$ while the public parameter is $pk = (u, v, g, spk)$.

SigGen(sk,F). The data owner chooses a random element *name* in Z_p as the name of file $F = \{m_i\}_{1\leq i\leq n}$ and computes the file tag as

$$t = name||Sig_{ssk}(name)$$

with signature on *name*. Then for each $m_i \in Z_p$, computes a signature σ_i as

$$\sigma_i = (H(i) \cdot u^{m_i})^x \in G(1 \leq i \leq n).$$

Finally, the data owner stores $\{F, \phi = \{\sigma_i\}_{1\leq i\leq n}, t\}$ on the cloud and deletes the files and its corresponding set of signatures from local storage.

Challenge(1^k)**.** When performing the auditing protocol, TPA retrieves the file tag t for F and checks the validity with *spk*, and quits if fail. If t is correct, TPA picks random c, k_1, k_2 in Z_p and sends $chal = (c, k_1, k_2)$ to the cloud server where k_1, k_2 are keys for pseudorandom permutation π and pseudorandom function f for each auditing task.

ProofGen$(F, \phi, chal)$**.** Upon receiving the challenge $chal$, the cloud server first determines the challenging subset $I = \{s_j\}(1 \leq j \leq c)$ of set $[1, n]$ by computing $s_j = \pi_{k_1}(j)$ and the corresponding coefficients by evaluating $v_{s_j} = f_{k_2}(j)(1 \leq j \leq c)$. For $i \in I$, the cloud server picks a random $r \in Z_p$, calculates $R = u^r \in G$ and $\mu = \mu^* + rh(R), \sigma = \prod_{i=s_1}^{s_c} \sigma_i^{v_i}$, where $\mu^* = \sum_{i=s_1}^{s_c} v_i m_i$. Finally, the cloud server sends the proof $P = (\mu, \sigma, R)$ to TPA.

VerifyProof$(pk, chal, P)$**.** Upon receiving the (μ, σ, R) from the server, TPA computes $s_j = \pi_{k_1}(j)$ and $v_{s_j} = f_{k_2}(j)(1 \leq j \leq c)$, and checks if

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{i=s_1}^{s_c} H(i)^{v_i} \cdot u^{\mu} \cdot R^{-h(R)}, v).$$

3 Security analysis of the scheme

A privacy-preserving public auditing scheme [Worku et al. 2014] should provide the properties of soundness and privacy. Thus, two kinds of adversaries are involved in this kind of protocols. The first one aims to attack the soundness while the second one tries to attack the privacy. In the following, we will discuss the security of the scheme in [Worku et al. 2014] under these attacks.

3.1 Soundness

A scheme is sound if any cheating prover that convinces the verification algorithm that it is storing a file is actually storing that file. Ateniese *et al.* [Atenisese et al. 2007] and Shacham-Waters [Shacham et al. 2008] formalized this notion by describing a security model for soundness. Worku *et al.*'s model [Worku et al. 2014] provides the adversary full access to the information stored on the cloud server. In the soundness game, the adversary is intended to play the role of a malicious cloud server without the challenged file, who interacts with a challenger playing the role of TPA. The security model for soundness described in [Worku et al. 2014] is as follows.

Setup. The challenger generates a keypair (pk, sk) by running KeyGen algorithm and provides pk to the adversary.

Phase 1. The challenger computes a signature for each block made by the adversary adaptively.

Challenge. The challenger challenges the adversary for proof and at the same time, interacts with normal execution protocol for data integrity check.

Phase 2. Phase 1 will be repeated for other blocks indices different from those already included in the challenge.

Output. The adversary outputs a proof that can pass the verification.

In the following, we show that with the help of signature queries, an adversary, i.e. a malicious cloud server, who has deleted the entire challenged file and the signatures corresponding to each block of this file, could generate a valid response without being detected by the TPA in the auditing process. The details are as follows.

Setup. The challenger generates a keypair (pk, sk) by running KeyGen algorithm and provides pk to the adversary.

Phase 1. The adversary makes two files $F = (m_1, m_2, \cdots, m_n)$ and $F' = (m'_1, m'_2, \cdots, m'_n)$, and then requests two signature queries on m_i and m'_i for any $1 \leq i \leq n$, and get two signatures σ_i and σ'_i.

Challenge. The challenger checks the integrity of his file, say $F^* = (m^*_1, \cdots, m^*_n)$ by sending a challenge $chal^* = (c^*, k^*_1, k^*_2)$. Note that this challenge on F^* never appeared in previous stages.

Phase 2. The adversary determines the challenging set $I^* = \{s^*_j\}$ by computing $s^*_j = \pi_{k^*_1}(j)$ for $(1 \leq j \leq c^*)$. Then for $1 \leq j \leq c^*$, the challenger makes a file in which the value of the s^*_j-th block is 0, and queries the signature of this block. Finally, the adversary obtains c^* signatures $\sigma^0_{s^*_j}$ for $1 \leq j \leq c^*$.

Output. The adversary generates a response to $chal^*$ in the following way.

1. Calculate the coefficient set of $v^*_{s_j}$ where $v^*_{s_j} = f_{k^*_2}(j)(1 \leq j \leq c^*)$.

2. Pick a random $r^* \in Z_p$, and calculate $R^* = u^{r^*} \in G$.

3. Choose c^* random elements $m^*_1, m^*_2, \cdots, m^*_{c^*}$ from Z_p.

4. Compute $\mu = \mu^* + r^* h(R^*)$, where $\mu^* = \sum_{j=1}^{c^*} v^*_{s_j} m^*_j$.

5. Compute $\sigma = \prod_{j=1}^{c^*} (\sigma^0_{s^*_j} \cdot [(\frac{\sigma_i}{\sigma'_i})^{\frac{1}{m_i - m'_i}}]^{m^*_j})^{v^*_{s_j}}$.

6. Send $P^* = (\mu, \sigma, R^*)$ to TPA.

The correctness of the forged response is illustrated below.

$$\begin{aligned}
e(\sigma, g) &= e(\prod_{j=1}^{c^*} (\sigma^0_{s^*_j} \cdot [(\frac{\sigma_i}{\sigma'_i})^{\frac{1}{m_i - m'_i}}]^{m^*_j})^{v^*_{s_j}}, g) \\
&= e(\prod_{j=1}^{c^*} \sigma^0_{s^*_j}, g)^{v^*_{s_j}} \cdot e(\prod_{j=1}^{c^*} (\frac{\sigma_i}{\sigma'_i})^{\frac{m^*_j}{m_i - m'_i}}, g)^{v^*_{s_j}} \\
&= e(\prod_{j=1}^{c^*} H(s^*_j), v)^{v^*_{s_j}} \cdot \prod_{j=1}^{c^*} e(\frac{\sigma_i}{\sigma'_i}, g)^{\frac{m^*_j v^*_{s_j}}{m_i - m'_i}} \\
&= e(\prod_{j=1}^{c^*} H(s^*_j)^{v^*_{s_j}}, v) \cdot \prod_{j=1}^{c^*} [\frac{e(H(i)u^{m_i}, v)}{e(H(i)u^{m'_i}, v)}]^{\frac{m^*_j v^*_{s_j}}{m_i - m'_i}} \\
&= e(\prod_{j=1}^{c^*} H(s^*_j)^{v^*_{s_j}}, v) \cdot \prod_{j=1}^{c^*} e(u^{m_i - m'_i}, v)^{\frac{m^*_j v^*_{s_j}}{m_i - m'_i}} \\
&= e(\prod_{j=1}^{c^*} H(s^*_j)^{v^*_{s_j}}, v) \cdot e(u^{\sum_{j=1}^{c^*} m^*_j v^*_{s_j}}, v) \\
&= e(\prod_{j=1}^{c^*} H(s^*_j)^{v^*_{s_j}}, v) \cdot e(u^{\mu - r^* h(R^*)}, v) \\
&= e(\prod_{j=1}^{c^*} H(s^*_j)^{v^*_{s_j}} u^{\mu} (R^*)^{-h(R^*)}, v)
\end{aligned}$$

In fact, the security model in [Worku et al. 2014] is unrealistic in the sense there is no a scheme that can be proven secure in this model. The reason is, the adversary is allowed to query signatures after he receives a challenge. Thus, the adversary could produce some random blocks and generate a valid response for these blocks as a valid response. The security proof in [Worku et al. 2014] describes only outside adversaries since signature queries are not involved. In addition, the cloud server can pollute stored files even in the correct security model, say the model due to Ateniese et al. [Atenisese et al. 2007, Ateniese et al. 2011], which does not contain the phase 2 queries. To be specific, in phase 1, the adversary requests two signature queries on m_i and m'_i, two blocks in two distinct files but with the same position i, and receives two signatures σ_i and σ'_i. Then the adversary computes $(\frac{\sigma_i}{\sigma'_i})^{\frac{1}{m_i - m'_i}}$ as the user's secret key u^x. Now the cloud server is able to modify any block m_k to $m_k + \Delta m_k$, and generate its signature by computing $\sigma_k \cdot (\frac{\sigma_i}{\sigma'_i})^{\frac{\Delta m_k}{m_i - m'_i}}$. With these values, the server can produce a valid response without original blocks.

3.2 Privacy

Regarding "data privacy", it seems that there is no a widely accepted notion to describe this property. The privacy notion is not being formally defined in [Worku et al. 2014] yet and from the part of privacy-preserving analysis in [Worku et al. 2014], we can see privacy here states that TPA cannot derive the entire file during the process of auditing. This property is true since μ^* is blinded by a random value r chosen by the server and keeps unknown to TPA, as $\mu = \mu^* + rh(R)$. This kind of privacy is similar to one-wayness of encryption and we argue that it is not strong enough in some scenarios say, in the context of dictionary attack. A recently proposed notion of "IND-privacy" [Fan et al. 2013] captures the essence of data privacy well in the sense that IND-privacy guarantees that TPA cannot obtain any information of the files via the integrity checking. Although it is claimed that "no information of μ^*" will be leaked to TPA in [Worku et al. 2014], in the following, we show that the scheme [Worku et al. 2014] cannot achieve IND-privacy.

We firstly review the security model of IND-privacy described in [Fan et al. 2013].

IND-Privacy. The data privacy for auditing proofs via an *indistinguishability* game between a simulator $\mathcal{S}$ (the prover) and an adversary $\mathcal{A}$ (the verifier).

Setup: The simulator runs KeyGen to generate (sk, pk) and passes pk to the adversary $\mathcal{A}$.

Phase 1: $\mathcal{A}$ is allowed to make signature queries. To make such a query, $\mathcal{A}$ selects a file F and sends it to $\mathcal{S}$. $\mathcal{S}$ generates a file tag t, signatures $\pi = \{\sigma_i\}$ for each block of F, and then returns (t, π) to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ chooses two distinct files F_0, F_1 that has not appeared in Phase 1, and sends them to $\mathcal{S}$. $\mathcal{S}$ calculates (t_0, π_0) and (t_1, π_1) by running the SigGen algorithm. $\mathcal{S}$ then tosses a coin $b \in \{0, 1\}$, and sends t_b back to $\mathcal{A}$. $\mathcal{A}$ generates a challenge *chal* and sends it to $\mathcal{S}$. $\mathcal{S}$ generates a proof $\mathcal{P}$ based on (F_b, t_b, σ_b) and $\mathcal{A}$'s challenge *chal* and then sends $\mathcal{P}$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a bit b' as the guess of b.

Define the advantage of the adversary $\mathcal{A}$ as

$$Adv_{\mathcal{A}}(\lambda) = |\Pr[b' = b] - 1/2|.$$

Definition 1. An auditing protocol has *indistinguishability* if for any polynomial time algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}(\lambda)$ is a negligible function of the security parameter λ.

Below we show that the scheme [Worku et al. 2014] cannot achieve indistinguishability. Let $\mathcal{A}$ denote an IND adversary which works as follows (see Fig. 1).

- $\mathcal{A}$ chooses two distinct files $F_0 = (m_{01}, \cdots, m_{0n})$ and $F_1 = (m_{11}, \cdots, m_{1n})$ such that $m_{0i} \neq m_{1i}$ for $1 \leq i \leq n$.
- $\mathcal{S}$ generates $(t_0, \{\sigma_{0i}\})$ and $(t_1, \{\sigma_{1i}\})$ for F_0 and F_1 respectively. $\mathcal{S}$ then chooses a random $b \in \{0, 1\}$ and sends t_b back to $\mathcal{A}$.
- After receiving the tag t_b, $\mathcal{A}$ chooses a random challenge $chal = \{i, \nu_i\}_{i \in I}$.
- $\mathcal{S}$ computes and sends to $\mathcal{A}$ the response $\mathcal{P} = (\mu, \sigma_b, R)$.
- $\mathcal{A}$ computes $\mu_0' = \sum_{i \in I}(\nu_i m_{0i})$ and checks if

$$e(\prod_{i \in I}(H(i))^{\nu_i} u^{\mu_0'}, v) = e(\sigma_b, g).$$

If it is true, return 0; otherwise, return 1.

Probability Analysis. If $b = 0$, then $\sigma_b = \sigma_0$ and the equation

$$e(\sigma_0, g) = e(\prod_{i \in I} H(i)^{v_i} u^{\mu_0^*}, v)$$

always holds. On the other hand, if $b = 1$, then $\sigma_b = \sigma_1$ and

$$e(\sigma_0, g) = e(\prod_{i \in I} H(i)^{v_i} u^{\mu_1^*}, v)$$

holds only when

$$\mu_0^* (= \sum_{i=s_1}^{s_c} v_i m_{0i}) = \mu_1^* (= \sum_{i=s_1}^{s_c} v_i m_{1i}),$$

which happens only with probability $1/p$ for randomly selected $\{\nu_i\}_{i \in I}$ since $m_{0i} \neq m_{1i}$ for all $i \in \{s_1, \cdots, s_c\}$. Therefore, $\mathcal{A}$ has an overwhelming probability to guess the value of b correctly.

4 Conclusion

In this paper, we analyzed the security of a privacy-preserving public auditing scheme [Worku et al. 2014] proposed recently and showed that it fails to achieve soundness in their security model and achieved limited privacy. Constructing public auditing protocols with perfect privacy-preserving is still worth the effort to put on in the near future.

$\mathcal{A}$		$\mathcal{B}$
Generate two distinct files:		
$F_0 = (m_{01}, \cdots, m_{0n})$		
$F_1 = (m_{11}, \cdots, m_{1n})$	$\xrightarrow{F_0,\ F_1}$	Compute $(t_0, \{\sigma_{0i}\})$ for F_0 and $(t_1, \{\sigma_{1i}\})$ for F_1
		Randomly choose $b \in \{0, 1\}$
	$\xleftarrow{t_b}$	
$chal = \{c, k_1, k_2\}$	$\xrightarrow{chal}$	Compute $s_j (1 \leq j \leq c)$
		$R = u^r$
		$\mu_b^* = \sum_{i=s_1}^{s_c} v_i m_{bi}$
		$\mu_b = \mu_b^* + rh(R)$
		$\sigma_b = \prod_{i=s_1}^{s_c} \sigma_{bi}^{v_i}$
	$\xleftarrow{\mu_b, \sigma_b, R}$	
Compute $\mu_0^* = \sum_{i=s_1}^{s_c} v_i m_{0i}$,		
Check if		
$e(\sigma_b, g) \stackrel{?}{=} e(\prod_{i \in I} H(i)^{v_i} u^{\mu_0^*}, v)$.		
If true, output 0; Otherwise, output 1.		

Figure 1: Indistinguishability analysis on the auditing protocol [Worku et al. 2014]

References

[Atenisese et al. 2007] Ateniese, G., Burns, R. C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z.N.J., Song, D.: "Provable data possession at untrusted stores"; Proc. of ACM CCS 2007, Alexandria, Virginia, USA, Oct.29-Nov.2, 2007, 598–609.

[Armbrust et al. 2010] Armbrust, M., Fox, A., Griffith,R., Joseph, A., Katz, R., Konwinski, A., Lee, G.: "A view of cloud computing"; Communications of the ACM, 53, 4, (2010) 50–58.

[Ateniese et al. 2011] 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. Security, 14, 1, (2011) 12.

[Boneh et al. 2001] Boneh, D., Lynn, B., Shacham, H.: "Short signatures from the weil pairing"; Proc. of ASIACRYPT 2001, Gold Coast, Australia, 2001, 514–532.

[Fan et al. 2013] Fan, X., Yang, G., Mu, Y., Yu, Y.: "On Indistinguishability in Remote Data Integrity Checking"; The Computer Journal, (2013) doi: 10.1093/comjnl/bxt137.

[Jules et al. 2007] Juels, A., Kaliski, B. S.: "PORs: proofs of retrievability for large files"; Proc. of ACM CCS 2007, Alexandria, Virginia, USA, Oct.29-Nov.2, 2007, 584–597.

[Shacham et al. 2008] Shacham,H., Waters, B.: "Compact proofs of retrievability"; Proc. of Asiacrypt 2008, Sydney, Australia, Jan. 8–11, 2008, 90–107.

[Shacham et al. 2012] Shacham,H., Waters, B.: "Compact proofs of retrievability"; Journal of Cryptology, 26, 3, (2013) 442–483.

[Wang et al. 2010] Wang, C., Ren, K., Lou, W., Li, J.: "Toward publicly auditable secure cloud data storage services", IEEE Network, 24, (2010) 19–24.

[Wang et al. 2010] Wang, C., Wang, Q., Ren, K., Lou, W.: "Privacy-preserving public auditing for data storage security in cloud computing"; Proc. of IEEE INFOCOM 2010, San Diego, CA, 14-19 March, 2010, 525–533.

[Wang et al. 2011] Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: "Enabling public audibility and data dynamics for storage security in cloud computing"; IEEE Trans. Parallel Distrib. Syst., 22, 2011, 847–859.

[Wang et al.2013] Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W.: "Privacy-preserving public auditing for secure cloud storage"; IEEE Transactions on Computers, 62, (2013) 362–375.

[Worku et al. 2014] Worku,S.G., Xu, C., Zhao, J., He, X.: "Secure and efficient privacy-preserving public auditing scheme for cloud storage"; Computers & Electrical Engineering, 40, 5, (2014) 1703–1713.

[Wei et al. 2014] Wei, L., Zhu, H., Cao, Z., Dong, X., Jia, W., Chen,Y. , Vasilakos, A.V.: "Security and privacy for storage and computation in cloud computing"; Information Sciences, 258, 10, (2014) 371–386.

[Yu, Niu et al. 2014] Yu Y., Niu, L., Yang G., Mu Y., Susilo W.: "On the security of auditing mechanisms for secure cloud storage"; Future Generation Comp. Syst. 30, (2014) 127–132.

[Yu, Ni et al. 2014] Yu Y., Ni J., Au M. H., Liu, H., Wang, H., Xu C.,: "Improved security of a dynamic remote data possession checking protocol for cloud storage"; Expert Syst. Appl. 41,17, (2014) 7789–7796.