On the Security of a Mutual Verifiable Provable Data Auditing in Public Cloud Storage

Jianhong Zhang[1,3], Pengyan Li[1], and Min Xu[2]
(Corresponding author: Jianhong Zhang)

College of Sciences & North China University of Technology, Beijing 100144, China[1]
Department of Education & Baoding University, Baoding 07100, China[2]
Guangxi Key Lab of Multi-source Information Mining & Security, Guilin, China[3]
(Email: xumin@163.com)

(Received Jan. 29, 2016; revised and accepted Apr. 23 & Sept. 24, 2016)

Abstract

Provable Data Possession (PDP) enables cloud users to verify the integrity of their outsourced data without retrieving the entire file from cloud servers. At present, to execute data checking, many PDP schemes is delegated to some proxy to implement remote data possession checking task. Because the proxy may store some state information in cloud storage servers, it makes that many PDP scheme are insecure. To solve this problem, Ren et al. proposed an mutual verifiable provable data possession scheme and claimed that their scheme is secure. Unfortunately, in this work, we show that their scheme is insecure. It exists forgery attack and replay attack. After giving the corresponding attacks, we give an improved scheme to overcome the above flaws. By analyzing, we show that our improved PDP scheme is secure under the Chosen-Target-CDH problem and the CDH problem.

Keywords: Diffie-Hellman Key Agreement; Forgery Attack; Mutual Verifiable; Provable Data Possession (PDP).

1 Introduction

Due to providing dynamically scalable resources provisioned as a service over the Internet, Cloud computing has been emerged as a new computing paradigm in Internet's evolution. Many advantages in Cloud computing draw people attentions [12]. These advantages make it has become an inevitable trend that individuals and IT enterprises store data remotely to the cloud in a flexible on-demand manner, namely cloud storage. As an important service of cloud computing, cloud storage allows the data owner to migrate his data files from their local disk to the remote cloud server, which is one of the most important cloud services. Meanwhile, it allows the cloud clients to flexibly access their outsourced files at anytime and from anywhere. For IT enterprises, it avoids the initial investment of expensive infrastructure setup, large equipment, and daily maintenance cost.

Although cloud storage can provide convenience for people, it also makes people face numerous security challenges since cloud server is not fully trusted [15]. Firstly, data owners would worry that their outsourced data could be revealed or accessed by the unauthorized users. Secondly, they would also worry that their outsourced data could be lost in the Cloud since cloud sever may discard the data which has not been accessed or rarely accessed to save the storage space or keep fewer replicas than promised. To the best of my knowledge, there exists a lot of incidents of data loss or leakage in recent years. For example, it was reported that "Amazon's huge EC2 cloud services crash permanently destroyed some data, and Hundreds of Dropbox Passwords are leaked. Based on the above worries, data owners hope a kind of cloud service to assure the integrity checking of the outsourced data.

To solve the above security challenging problem, Ateniese et al. [1] proposed the model of Provable Data Possession (PDP) to solve the storage auditing problem by spot-checking technique. And gave two provably secure and practical PDP schemes from the RSA assumption in 2007. They can achieve the data integrity checking of static data. In the same year, Juels et al. [6] proposed the notion of Proof of Retrievability (PoR), in which the integrity of the data file is completed by checking the correctness of the inserted sentinel blocks. The number of times that the file can do integrity verification is limited since the sentinel blocks are one-time labels. Subsequently, Shacham and Waters [10] put forward two efficient and compact PoR schemes. The first one which is based on BLS signature, is publicly verifiable; the second one which is based on pseudo-random functions (PRFs), only supports private verification.

According to the roles of the verifier, the PDP protocols are divided into two categories: private PDP and public PDP. In some cases, private PDP is very necessary. For example, in Cloud-EHR (Electronic Health Records), the patient only hopes his own or the designated doctors to check the integrity of health records in case that these sensitive information is leaked. Recently Shen and Tzeng proposed a delegable provable data possession scheme [11], in which data owner produces the delegation key for delegated verifier and stores the key in cloud storage service (CSS) for verification. In [14], Wang et al. proposed a proxy provable data possession (PPDP) model. In PPDP, data owner is able to delegate a proxy to enforce its remote data possession checking on behalf of his own. Unfortunately, the two private PDP schemes are shown to be insecure since the state information of the proxy or delegated verifier is controlled by a malicious CSS [?].

With the proliferation of cloud storage, a variety of cloud auditing protocols and their variants [2, 3, 4, 5, 7, 8, 13, ?, 17, 18] were proposed for catering some specific properties, such as public verification, dynamic operations and privacy preserving. According to the different requirements of the stored data and various properties of cloud storage services, A majority of the storage services are provided. For example, Spideroak [2], it is a good solution for security and privacy, which might be particularly appealing if a user would like to store sensitive data; Dropbox [2, 16], as one of the first online storage services, is useful, reliable and works across multiple platforms.

Recently, to overcome the above problems in private PDP, Ren et al. proposed an efficient mutual verifiable provable data possession (MV-PDP) scheme by using Diffie-Hellman key agreement. In their MV-PDP scheme, the verifier is stateless and independent from CSS, thus, the scheme efficiently overcomes the problem which the verifier can be optionally specified by a malicious CSS. Very regretfully, in this work, we show that Ren et al's MV-PDP scheme is insecure, it exists forgery attack, replay attack and the deleting attack of malicious cloud server. After we give the corresponding attacks, the reasons to produce such attacks are analyzed. Finally, to overcome our attacks, an improved PDP scheme is proposed. And the improved scheme is shown to be secure under the Chosen-Target-CDH problem and the CDH problem.

2 Preliminaries

2.1 System Model

For a PDP scheme, it involves three entities: cloud client, cloud storage server and the verifier. The cloud server has huge storage space and computational resources. It provides data storage service and charges cloud users according to pay-per-use regulation. In general, cloud users are the resource-constrained devices, it needs to rent data storage service and interact with the cloud server to upload, access and update their stored data. The verifier may be a cloud user or a delegated party.

2.2 Computational Assumptions

The security of the improved signature schemes is based on the hardness of the CDH problem for the scheme in [?]. The Chosen-Target-CDH problem is defined as follows: the solver S receives as input a pair (g, g^a), where g is a generator of G_1 with the prime order q, and $a \in Z_q$ is a random value. The solver S has adaptive access to two oracles:

1) Target oracle: this oracle outputs a random element $Z_i \in G$;

2) Helper oracle: this oracle takes as input an element $W_i \in G$ and outputs the element W_i^a.

We say that the solver $S(q_t, q_h, d)$-solves the Chosen-Target-CDH problem, for $q_t \geq d > q_h$, if it makes q_t and q_h queries, respectively, to the target and helper oracles, and after that it outputs d pairs $((V_1, j_1), \cdots, (V_d, j_d))$ such that:

1) All the elements V_i are different;

2) For all $i \in \{1, 2, \cdots, d\}$, the relation $V_i = Z_{ji}^a$ is satisfied where Z_{ji} is the element output by the target oracle in the $ji-$th query.

3 Reviews of Ren et al.'s Public Auditing Scheme

Recently, to overcome the proxy which stores some state information in cloud storage servers, Ren et al. proposed a mutual verifiable public auditing scheme. In their scheme, the verifier is stateless and independent of cloud server, and the scheme is very efficient in terms of auditing cost. In the following, we briefly review Ren et al.'s scheme. Please the interested reader refer to [9] for the detail.

Setup: Let λ be a security parameter, taking λ as an input, output the following parameters. G is a cyclic multiplicative group with the large prime order q. g is a generator of group G. Choose two hash functions H_1 and H_2, which satisfy $H_1 : \{0,1\}^* \rightarrow Z_q$ and $H_2 : \{0,1\}^* \rightarrow Z_q$. The system parameters are

$$(g, G, q, H_1, H_2).$$

KeyGen: For a client, it chooses a random number $x \in Z_q$ as private key and produces public key $X = g^x$. And the client designates a trust verifier DV to enforce integrity checking. At the same time, DV runs **KeyGen** to produce a public-private pair $(y, Y = g^y)$.

TagGen: Assume that the outsourced file F is divided into n blocks $\{m_1, m_2, \cdots, m_n\}$, to produce authentication tag on data block m_i, it computes as follows:

1) The client computes $k_{i1} || k_{i2} = H_1(Y^x, m_i)$;

2) Then it computes

$$\delta_{i1} = (Y^{H_2(m_i)k_{i1}+k_{i2}})$$
$$\delta_{i2} = X^{k_{i1}}$$
$$\delta_{i3} = X^{k_{i2}}.$$

3) Finally, the authentication tags are denoted as $\phi = (\delta_{i1}, \delta_{i2}, \delta_{i3})$, $i \leq i \leq n$.

The client sends (F, ϕ) to the cloud server and deletes them form it local disk.

Challenge phase: In their protocol, only the client or the designated verifier (DV) is able to verify the integrity checking of the outsourced data with the cloud server. To execute the integrity checking, the verifier randomly chooses a c-element subset I of the set $[1, n]$, and for $i \in I(1 \leq i \leq c)$, it selects random element $v_i \in Z_q$. Finally, the verifier sends the challenge information $chall = \{(i, v_i)\}_{i\in I}$ to the cloud server.

GenProof: On receiving the challenge information, the cloud server computes as follows.

$$\sigma = \sum_{i=1}^{c} \delta_{i1}^{v_i}$$
$$\delta = \sum_{i=1}^{c} \delta_{i2}^{H_2(m_i)v_i}$$
$$\eta = \sum_{i=1}^{c} \delta_{i3}^{v_i}.$$

The cloud server outputs the proof information $pf = (\sigma, \delta, \eta)$ to the verifier as the corresponding response.

VerifyProof: After receiving the returned proof information $pf = (\sigma, \delta, \eta)$ the designated verifier checks whether the following equation holds by using public key X and his private key y.

$$\sigma = (\delta \cdot \eta)^y$$

If the equation holds, then it outputs true, otherwise, outputs false.

4 Security Analysis of Ren et al.'s Scheme

In [1], by using the Diffie-Hellman agreement key, Ren et al. proposed an auditing protocol in public cloud. Their protocol only supports the verification of the client and the designated verifier, and they also claimed that their scheme is secure, the security of their scheme is based on the CDH problem. Unfortunately, by analyzing the security of the scheme, we show that their scheme is insecure. Any one can produce a forged proof information to make that the designated verifier ensures the returned response satisfies the verification equation. That is to say, the designated verifier's integrity checking cannot guarantee the security of outsourced data in cloud server .

In the following, we give the corresponding attack and analyze the reason to produce such attack.

4.1 Forgery Attack

Let A be an adversary, to forge a false proof information, it randomly chooses $R_1 \in G$ and $r \in Z_q$ to set

$$\begin{aligned} \delta^* &= R_1 \\ \eta^* &= R_1^{-1} \cdot g^r \\ \sigma^* &= Y^r. \end{aligned}$$

Thus, the forged proof information is $(\delta^*, \eta^*, \sigma^*)$.

In the following, we show the forged proof information $(\delta^*, \eta^*, \sigma^*)$ can pass the verification of the designated verifier. since the designated verifier can use its private key y to compute

$$\begin{aligned} (\eta^* \delta^*)^y &= (R_1 \cdot R_1^{-1} \cdot g^r)^y \\ &= (g^{ry}) \\ &= Y^r \\ &= \sigma^* \end{aligned}$$

It means that the forged proof information $(\delta^*, \eta^*, \sigma^*)$ satisfies verification equation. Thus, our attack is valid.

The reason to produce such attack is that each component in the proof information $pf = (\sigma, \delta, \eta)$ is free, the relation of each component with the other ones among the proof information cannot be constrained each other. At the same time, for each challenge, the challenge information has not shown in the verification equation, it gives the attacker to provide a forgery chance. To overcome such attacker, the relation of elements in the proof information must be restrained.

4.2 Replay Attack of Cloud Server

In cloud storage, cloud server is a un-trusted entity. Here, we will show that cloud server how to implement replay attack.

Let $pf = (\sigma, \delta, \eta)$ be a valid proof information which is from a challenge. When the designated verifier makes a new challenge with cloud server. Cloud server randomly chooses $r \in Z_q$ to compute

$$\begin{aligned} \delta^* &= \delta^r \\ \eta^* &= \eta^r \\ \sigma^* &= \sigma^r \end{aligned}$$

Then, it sets $prf^* = (\delta^*, \eta^*, \sigma^*)$ as a proof information and returns it to the designated verifier.

In the following, we show that the returned proof information can pass the verification of the designated verifier.

Since

$$\begin{aligned}(\delta^* \cdot \eta^*)^y &= (\delta^r \cdot \eta^r)^y \\ &= (\delta^r \cdot \eta^r)^y \\ &= (\delta \cdot \eta)^{y \cdot r} \\ &= \sigma^r \\ &= \sigma^*\end{aligned}$$

Obviously, it satisfies the verification equation of the designated verifier. It means that cloud server's replay attack is valid.

The reason to produce such attack is that verification equation is independent of each proof information. And verification equation has homomorphism. Thus, it is very easy to result in replay homomorphism attack.

4.3 Delete Attack

According to Ren et al.'s scheme, in the VerifyProof phase and GenProof phase, the stored message F is not used. Thus, the malicious cloud server can delete the stored message F, it only keeps all block's authentication tags $\phi = (\delta_{i1}, \delta_{i2}, \delta_{i3})$, $1 \leq i \leq n$ and each block's hash value $H_2(m_i), 1 \leq i \leq n$. Then the cloud server can produce a valid proof information by all $H_2(m_i)$ and ϕ. Thus, their scheme exists the cloud server's deleting attack.

5 An Improved Auditing Scheme in Public Cloud

To overcome the above attacks, based on Diffie- Hellman key agreement idea, we give an improved auditing scheme. The details are as follows:

Setup: Let λ be a security parameter, taking λ as an input, output the following system parameters. G is a cyclic multiplicative group with the large prime order q. g is a generator of group G. h is a random element in group G, and select three hash functions $H_)$, H_1 and H_2, which satisfy $H_0 : \{0,1\}^* \rightarrow G$,$H_1 : \{0,1\}^* \rightarrow Z_q$ and $H_2 : \{0,1\}^* \rightarrow Z_q$. Let $f : \{0,1\}^k \times \{0,1\}^{log_2 n}\{0,1\}^l$ denotes a pseudo-random function and $\pi : \{0,1\}^k \times \{0,1\}^{log_2 n} \rightarrow \{0,1\}^{log_2 n}$ represents a pseudo-random permutation. The system parameters are

$$(g, h, G, q, \pi, f, H_0, H_1, H_2)$$

KeyGen: In this phase, KeyGen is the same as that of Ren et al's scheme.

TagGen: Let F denote the outsourced file, it is separated into n blocks $\{m_1, m_2, \cdots, m_n\}$, to produce authentication tag on data block m_i, it computes as follows:

1) The client computes $k_{i1}||k_{i2} = H_1(Y^x, m_i)$;

2) Then it computes

$$\begin{aligned}\delta_{i1} &= H_0(name, i, Y^x)Y^{xm_i}(Y^{H_2(m_i)k_{i1}+k_{i2}})^x \\ \delta_{i2} &= X^{k_{i1}} \\ \delta_{i3} &= X^{k_{i2}}\end{aligned}$$

3) Finally, the authentication tags are denoted as $\phi = (\delta_{i1}, \delta_{i2}, \delta_{i3})$, $i \leq i \leq n$.

The client sends (F, ϕ) to the cloud server and deletes them form it local disk.

Challenge phase: The same as Ren et al.'s protocol, in our improved protocol, only the client or the designated verifier (DV) is allowed to verify the integrity checking of the outsourced data. To achieving the integrity checking, the verifier randomly chooses an integer c and two keys k_1 and k_2 for f and π respectively. Then, the verifier sends the challenge information $chall = (c, k_1, k_2)$ to the cloud server.

GenProof: Upon receiving the challenge information $chall = (c, k_1, k_2)$, the cloud server computes as follows.

1) For $j = 1$ to c, it computes $i_j = \pi_{k_2}(j)$ as the challenged blocks' indices, and it computes $a_j = f_{k_1}(j)$ as random coefficients.

2) Then compute

$$A = \sum_{i=1}^{c} a_i m_i$$

;

3) Next it computes

$$\sigma = \sum_{i=1}^{c} \delta_{i1}^{v_i}$$

$$\delta = \sum_{i=1}^{c} \delta_{i2}^{H_2(m_i)v_i}$$

$$\eta = \sum_{i=1}^{c} \delta_{i3}^{v_i}.$$

The cloud server outputs the proof information $pf = (D = X^A, \sigma, \delta, \eta)$ to the verifier as the corresponding response.

VerifyProof: After receiving the returned proof information $pf = (D, \sigma, \delta, \eta)$ the designated verifier executes as follows:

1) It firstly produces the challenged blocks i and the corresponding coefficients a_i.

2) Then, it computes

$$R = \prod_{i=1}^{c} H_0(i, X^y)^{a_i}$$

.

3) It checks whether the following equation holds by using public key X and his private key y.

$$\sigma \cdot X^{-yA} = (\delta \cdot \eta)^y \cdot R.$$

If the equation holds, then it outputs true, otherwise, outputs false.

6 Security Analysis of the Improved Scheme

In the section, we show that our improved scheme can achieve the properties of completeness, soundness and data privacy.

6.1 Completeness

For the client and the cloud server, if both of them are honest, then for each authentication tag $(\delta_{i1}, \delta_{i2}, \delta_{i3})$ and the challenged information (c, k_1, k_2), the completeness of the protocol is demonstrated as follows.

$$\sigma \cdot X^{-yA} = \prod_{i=1}^{c} \delta_{i1}^{a_i} \cdot (X^y)^{-A}$$
$$= (\prod_{i=1}^{c} H_0(name, i, Y^x) Y^{xm_i} (Y^{H_2(m_i)k_{i1}+k_{i2}})^x)^{a_i}$$
$$\cdot (X^y)^{-A}$$
$$= (\prod_{i=1}^{c} H_0(name, i, Y^x))^{a_i} \prod_{i=1}^{c} (Y^{H_2(m_i)k_{i1}+k_{i2}}))^{a_i x}$$
$$= (\prod_{i=1}^{c} H_0(name, i, Y^x))^{a_i} \prod_{i=1}^{c} (X^{H_2(m_i)k_{i1}+k_{i2}}))^{a_i y}$$
$$= (\delta \cdot \eta)^y \cdot R.$$

6.2 Soundness

Theorem 1. *If there exists a malicious cloud server can produce a false proof information to convince the designated verifier that the outsourced data is integrity in our improved scheme, then the CDH problem in group G can be solved.*

Proof. We will show for any PPT adversary A who wins the soundness of the game, there exists a challenger B that can construct a simulator S to solve an instance of the CDH problem.

Setup: Let (g, g^a, g^b) be an instance of the CDH problem. B sets (G, g, q) as system parameters. Let the attacked client's public key be pk_c and set $pk_c = g^a$. And B randomly choose $r \in Z_q$ as private key of the designated verifier and computes its public key $pk_v = g^r$.

Queries: A can adaptively make the following queries H_0-Oracle, H_1-Oracle,H_2-Oracle and TagGen-oracle during the execution. B responses these oracles as follows.

$H_1 - Oracle$**:** When an adversary A makes a H_1-query with $(*, m_i)$, it the record $(*, m_i)$ exists, then it outputs (k_{i1}, k_{i2}). Otherwise, it tosses a coin with the probability $Pr[coin_i = 1] = \zeta$ and randomly chooses $y_i \in Z_q$ to answer the following query.

1) If coini=0, then B sets $Y^{H_2(m_i)k_{i1}+k_{i2}} = g^b$; and insert $(*, m_i, g^b, \perp, \perp, coin_i)$ in the H_1-list which is initially empty.

2) If coini=1, then B sets $Y^{H_2(m_i)k_{i1}+k_{i2}} = g^{y_i}$; and insert $(*, m_i, g^{y_i}, k_{i1}, k_{i2}, coin_i)$ in the H_1-list, where k_{i1}, k_{i2} are two random numbers.

H_2**-Oracle:** When an adversary A queries the H_2-oracle with m_i. If the record (m_i, χ_i) exists in the H_2-list which is initially empty, then χ_i is returned. Otherwise, B randomly selects $\chi_i \in Z_q$ and returns it to A. And insert (mi, χ_i) in the H_2-list.

TagGen Oracle: When the adversary A makes a TagGen oracle query with m_i. If coini=0 which corresponds to m_i in the H_1-list, then it aborts it. Otherwise, if m_i does not exist in the H_1-list, then it randomly chooses $k_{i1}, k_{i2} \in Z_q$ to compute

$$\begin{aligned} \delta_{i1} &= (pk_c^{H_2(m_i)k_{i1}+k_{i2}})^r \\ \delta_{i2} &= pk_c^{k_{i1}} \\ \delta_{i3} &= pk_c^{k_{i2}}. \end{aligned}$$

If m_i exists in the H_1-list, then it uses the returned k_{i1}, k_{i2} to produce authentication tag. Finally, B returns $(\delta_{i1}, \delta_{i2}, \delta_{i3})$ to the adversary A.

Forgery: The adversary A outputs the forgery authentication tag on message m^* as follows:

$$(m^*, \delta_{i1}^*, \delta_{i2}^*, \delta_{i3}^*).$$

If the verification fails or the $coin_i^* = 0$ which corresponds to m^* in the H_1-list, then B claims it is failure. Otherwise,

$$\begin{aligned} \delta_{i1}^* &= (pk_v^{H_2(m^*)k_{i1}^*+k_{i2}^*})^a \\ &= (g^{H_2(m^*)k_{i1}^*+k_{i2}^*})^a \\ &= g^{ab} \end{aligned}$$

It means that the CDH problem can be solved . Obviously, it is in contradiction with the difficulty of solving the CDH problem.

□

Theorem 2. *For the cloud server, if it can return a false proof information which passes the verification of the designators, then the Chosen-Target-CDH problem is able to be solved.*

Proof. Assume that there exists a probabilistic adversary A which breaks our improved mutual PDP scheme, then we are able to construct a solver B of the Chosen -Target-CDH problem, which uses A to solve the Chosen- Target-CDH problem.

First of all, B initializes A by setting up the system parameters. Therefore, the solver B chooses a group G_1 with prime order q. g is a generator of group G_1.

The solver B asks for an instance of the Chosen-Target-Inverse-CDH problem in the group G_1. It receives 2-tuple (g, g^a) for some random secret number $a \in Z_q$. And it is allowed to access the target oracles and the helper oracles.

In the following game, B randomly chooses $k \in Z_q$ to compute $X = g^k$ as the public key of the challenged client. And set $Y = g^a$ as the public key of the designated verifier.

Queries: Once A is started with public parameters and public keys X, Y as input, a series of following queries may occur, where H_1-Oracle, H_2-Oracle, and TagGen Oracle are the same as those of Theorem 1.

Proof-Oracle: In this query, the challenger C behaves as the verifier and the adversary acts as the prover and query the solver B. And C makes at most q_p proof queries.

C makes a challenge information $chall = (c_i, k_{i1}, k_{i2})$ to the adversary A, then the adversary A does as follows.

1) For $j = 1$ to c_i, it computes $i_j = \pi_{k_{2i}}(j)$ as the challenged block's indices, and it computes $a_{ji} = f_{k_{1i}}(j)$ as random coefficients. And it computes

$$R_i = \prod_{j=1}^{c} H_0(j, X)^{a_j}$$

where $H_0(j, X)$ has already been queried by the adversary.

2) Then it queries the solver B for a random element, and B makes a query to its target oracle, and receives a random element $S_i \in G_1$ as answer from its target oracle.

3) Subsequently, it randomly selects $r \in Zq, Q_i \in G_1$ and sets $\delta_i = Q_i, D_i = g^r$ and $\eta_i = S_i/(Q_i \cdot D_i)$.

4) Next the adversary sends S_i to the solver B, at the same time, the solver B sends to the helper oracle the value S_i, and obtains as answer the value $T_i = S_i^a$. Finally, return it the adversary A.

5) Finally, it sets $\sigma_i = T_i/R_i$ and returns proof information $(D_i, \delta_i, \eta_i, \sigma_i)$.

Output: Eventually, the adversary outputs a proof information $(D_i^*, \delta_i^*, \eta_i^*, \sigma_i^*)$, it should satisfies

$$\sigma^* = (\delta^* \cdot \eta^* \cdot D^*)^y \cdot R^*.$$

Obviously, (S_i, T_i) and $(S^* = \delta^* \cdot \eta^* D^*, T^* = \sigma^*/R^*)$ are different. It means that the solver B can output it outputs q_p+1 pairs (S_i, T_i), however, it only makes q_p target oracle queries. Due to difficulty of solving the Chosen-Target-CDH, our improved PDP scheme is secure.

□

7 Performance Analysis

In this section, we evaluate the performance of the improved scheme in terms of computational cost. In cloud storage, data owner is a resource-restricting entity and the auditor is a very resource demanding service entity in terms of computational resource. Their computation efficiency has very important influence on the whole system. Therefore, we simulate the computational cost of the data owner and the verifier on a Mac OS X system with an Intel Core 2 Duo CPU at 2.5 GHz and 4.00-GB RAM. The code uses the pairing-based cryptography library version 0.5.12 to simulate the improved scheme.

7.1 Computational Cost of Data Owner

In our scheme, computational cost of data owner is determined by the number of producing tags for data block. For a constant size data file M, the number of data block is computed as $n = \frac{sizeof(M)}{sizeof(m_{block})}$, where $sizeof(m_{block})$ denotes the length of each data block. When we consider the time of producing a tag for one data block, it is easy to see that the computational time can be denoted as

$$Time_{tag} = 5C_{Mul} + 2C_H$$

where the symbols C_H and C_{Mul} denote hash operation which map to point of group G and point multiplication.

The total tag generation time for a constant size of data M can be computed as

$$TTime_{tag} = \frac{sizeof(M)}{sizeof(m_{block})}(2C_H + 5C_{Mul}). \quad (1)$$

According to Equation (1), we know the size of data block has influences on computation time of data owner. Figure 1 shows the total computation time of generating all the data tags for 1 MByte data component versus the size of each data block.

7.2 Computational Cost of the Verifier

In our scheme, the computational cost of the verifier includes the verification of data integrity checking equation. And it is linear to the number c of the challenged data

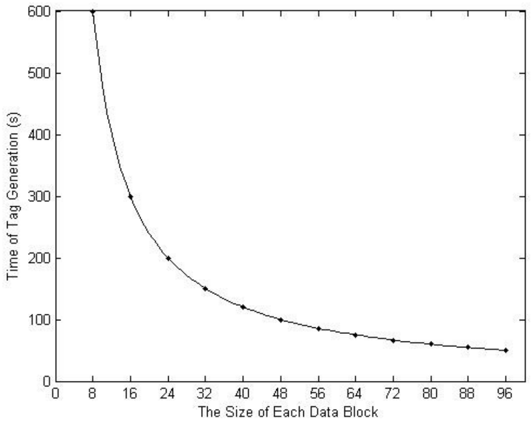

Figure 1: Impact of data block's size on computation time of data owner

blocks. It is easy to see that the computational time can be described as

$$Time_{ver} = (c)C_H + (c+2)C_{Mul} + 3C_e.$$

According to the above equation, we know that computation cost of the verifier is influenced by the challenged block's number. However, the time-consuming pairing operation is required in our improved scheme.

8 Conclusions

In this paper, we analyze the security of Ren et al.'s mutual PDP scheme, and show that their scheme is insecure against forgery attack, replay attack and deleting attack. It does not satisfy the security which are claimed in their scheme. After analyzing the reasons to produce such attacks, we proposed an improved PDP scheme to overcome the attacks which we launch on their scheme. Finally, we also analyze the security of the improved PDP scheme, and show that it is secure under the Chosen- Target-CDH problem.

Acknowledgments

This work is supported by Beijing Natural Science Foundation (No:4162020), subject construction of North China University of Technology (NO:XN-083) and Research Fund of Guangxi Key Lab of Multi-source Information Mining & Secu-rity (No. MIMS16-01).

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," *ACM Conference on Computer and Communications Security*, pp. 598–609, 2007.

[2] S. Cornelius, "Best online backup: Backblaze vs crashplan vs carbonite vs dropbox vs sugarsync vs crashplan vs spideroak & more!," May 17, 2016. (`http://www.werockyourweb.com/best-online-backup/`)

[3] W. Fu Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for secure data storage in cloud computing," *International Journal of Network Security*, vol. 18, no. 1, pp. 133–142, 2016.

[4] M. S. Hwang, C. C. Lee, T. H. Sun, "Data error locations reported by public auditing in cloud storage service," *Automated Software Engineering*, vol. 21, no. 3, pp. 373–390, 2014.

[5] M. S. Hwang, T. H. Sun, C. C. Lee, "Achieving dynamic data guarantee and data confidentiality of public auditing in cloud storage service," *Journal of Circuits, Systems, and Computers*, vol. 26, no. 5, 2017. (DOI: 10.1142/S0218126617500724)

[6] A. Juels, S. Burton, and Jr. Kaliski, "PORS: Proofs of retrievability for large files," in *ACM Conference on Computer and Communications Security*, pp. 584–597, Nov. 2007.

[7] C. W. Liu, W. Fu Hsien, C. C. Yang, and M. S. Hwang, "A survey of attribute-based access control with user revocation in cloud data storage," *International Journal of Network Security*, vol. 18, no. 5, pp. 900–916, 2016.

[8] C. W. Liu, W. Fu Hsien, C. C. Yang, and M. S. Hwang, "A survey of public auditing for shared data storage with user revocation in cloud computing," *International Journal of Network Security*, vol. 18, no. 4, pp. 650–666, 2016.

[9] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," *Journal of Internet Technology*, vol. 16, no. 2, pp. 317–323, 2015.

[10] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology*, vol. 26, no. 3, pp. 442–483, 2013.

[11] S. T. Shen and W. G. Tzeng, "Delegable provable data possession for remote data in the clouds," *Proceedings of ICICS'11*, pp. 93–111, 2011.

[12] J. Singh, "Cloud based technique for Blog search optimization," *International Journal of Electronics and Information Engineering*, vol. 4, no. 1, pp. 32–39, 2016.

[13] M. Stanek, "A note on security protocol for multicast communications," *International Journal of Network Security*, vol. 14, no. 1, pp. 59–60, 2012.

[14] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, 2016.

[15] Z. Wang, Y. Lu, G. Sun, "A policy-based deduplication mechanism for securing cloud storage," *International Journal of Electronics and Information Engineering*, vol. 2, no. 2, pp. 70–79, 2015.

[16] Q. Q. Xi, S. R. Jiang, L. M. Wang, and C. C. Chang, "Composable secure roaming authentication protocol for cloud-assisted body sensor networks," *International Journal of Network Security*, vol. 18, no. 5, pp. 816–831, 2016.

[17] J. Zhang and Q. Dong, "Efficient id-based public auditing for the outsourced data in cloud storage," *Information Sciences*, vol. 343-344, no. 2, pp. 1–14, 2016.

[18] J. Zhang, P. Li, and J. Mao, "Ipad: Id-based public auditing for the outsourced data in the standard model," *Cluster Computing*, vol. 19, no. 1, pp. 127–138, 2016.

Biography

Jianhong Zhang received his Ph.D. degrees in Cryptography from Xidian University, Xian, Shanxi, in 2004 and his M.S. degree in Computer Software from Guizhou University, Guiyang, Guizhou, in 2001. He was engaging in postdoctoral research at Peking University from October 2005 to December 2007. He has been an Assistant Processor of College of Sciences, North China University of Technology, Beijing China, since 2001. His research interests include computer networks, cryptography, electronic commerce security, computer software.

Pengyan Li received the BS. in Mathematics and Educa- tion from Luoyang Normal University in 2015. She is now a master in College of Science, North China University of Technology. She has published 2 papers in international conferences and journals. His research interest includes applied mathematics, education theory and multimedia processing..

Min Xu received the MS. in School of Electronic and Information Engineering from Hebei Normal University in 1998. She is now an Associate Professor in Department of Education, Baoding University. She has published more than 30 papers in international conferences and journals. His research interest includes applied cryptography, web security, computer software and multimedia processing.