TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 12, No. 9, September 2014, pp. 7007 ~ 7013
DOI: 10.11591/telkomnika.v12i9.4876

A Double-Efficient Integrity Verification Scheme to Cloud Storage Data

Deng Hongyao*[1,2], Song Xiuli[3], Tao jingsong[4]
[1]School of Mathematics & Computer Science, Yangtze Normal University,
No. 98 Julong Rd., Lidu Fuling District, Chongqing, 408000, China. Ph: +86 18996411262
[2]School of Information & Software Engineering, University of Electronic Science and Technology of China,
No. 2006, Xiyuan Ave., West Hi-Tech Zone, Chengdu, 611731, Sichuan China, Ph/Fax: +86 28 83201864
[3]Dept. of Computer Science and Technology, Chongqing University of Posts and Telecommunications,
No. 2 Chongwen Rd., Nan-an District, Chongqing, 400065, China, Ph: +86 23 62461404
[4]School of Electrical Engineering, Wuhan University,
Luojiashan,Wuchang, Wuhan,430072, China, Ph: +86 27 68770776
*Corresponding author, e-mail: hydeng_2004@163.com[1], songxl@cqupt.edu.cn[2], jamson_tao@163.com[3]

Abstract

This paper proposed two integrity verification schemes based on Schnorr Signature Scheme. One is safety integrity verification scheme (SIVS). Another is efficient integrity verification scheme (EIVS). They are difference in characteristics. EIVS has good computational costs while SIVS has high security guarantee. However, they are similar in work. The cloud storage server will choose a set of file blocks and verification blocks randomly while the user sends a challenge, and generate response values to send to the user. The user generates a set of signatures to verify the values. The aim is to check whether the cloud storage server preserves perfectly the user's file or not. In contrast with other schemes, the approach not only has double integrity verification guarantee schemes but also pay lower costs for communication and computation.

Keywords*: cloud storage, Schnorr signature scheme, double-efficient verification, challenge & response*

1. Introduction

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Cloud storage is a model of networked online storage where data is stored in virtualized pools of storage which are generally hosted by third parties. Hosting companies operate large data centers, and users who require their data to be hosted buy or lease storage capacity from them[2]. When the users store their data to cloud storage servers, they will gain a real convenience and save more investment. However, the users might lost control on their own data, and they might be unclear which device their data are stored at, so they are greatly concerned about the integrity of these data in cloud storage servers. How to verify remote data possession and data integrity, many scholars have carried out relevant research.

In 2003, Deswarte et al. [3] proposed a data integrity verification scheme based on RSA algorithm. The scheme, intending to check the integrity of remote data, performed exponentiation operation on entire document based on RSA algorithm. Ateniese et al. [4] built upon a Provable Data Possession (PDP) model to allow the user to utilize RSA-based homomorphic tags to challenge the server, which selected randomly data blocks and tags to generate the proofs to prove he stored intact the user's data. In a subsequent work, Curtmola et al. [5] proposed a multiple replica PDP (MR-PDP) scheme. This scheme ensured that multiple replicas of the user's data were stored at the untrusted storage server. In [6], Chen used homomorphic hash to present another PDP method, and the user's data would be preserved well in cloud storage server by the method. Juels et al.[7] described a Proof of Retrievability (PoR) model, and this model used spot-checking and error-correcting codes to ensure both "possession" and "retrievability" of the files on remote servers. In [8], the scheme allowed the

Received October 22, 2013; Revised June 27, 2014; Accepted July 21, 2014

verifier challenge the server without limit, and the server converge the tags of all data blocks into a short tag that would be taken as response message to the verifier. In [12], Cong Wang et al. utilized the homomorphic token to ensure the integrity of erasure-coded data with additional feature of data error localization. In a subsequent work, Qian Wang et al. [13] allowed a third party auditor to verify the integrity of the data stored in cloud based on Merkle hash tree.

Unfortunately, the computational complexity of above some schemes is too height, such as [3], some major cause lay in the fact that the server must exponentiate the entire file and access all of the file's blocks. The challenge number of some schemes is limit, such as [7]. To each check, "sentinel" must be disclosed to the server, and the verifier can't use again leaked "sentinel". Scheme [4] and [8] remedy partially some drawback of scheme [3] and [7] as the server selected randomly some of data blocks to achieve the probability of successful verification. The two schemes lower the computational requirements for the server and allow the verifier challenge the server without limit, but the time generating authentication tags is too long. This paper will propose two integrity verification schemes to improve above schemes in computational complexity.

2. Schnorr Signature Scheme

Schnorr signature scheme was proposed by Claus P. Schnorr, and it was patented in 1991 [9]. In order to describe our integrity verification schemes better, we adjusted the values of the parameters of Schnorr Signature Scheme.

Supposed p and q are two big prime, and $p-1$ is a multiple of q; g is a generator of Z_p^*, and $g^q = 1 \bmod p$; x is a private key of Z_q^*; y is a public key and $y = g^x \bmod p$; $h(\cdot)$ is an approved cryptographic hash function. If a signer signs the message m, then he chooses randomly a secret number $r \in Z_q^*$ and computes

$$u = g^r \left(\bmod p\right), e = h\left(m \| u\right), s = xe + r\left(\bmod q\right) \quad (1)$$

The signer sends the message m and the signatures (e,s) to the receiver. If the receiver has received m and (e,s), in order to verify the validity of the signature, then he first computes

$$u' = g^s y^{-e} (\bmod p) \quad (2)$$

And checks the following equation:

$$e \stackrel{?}{=} h\left(m \| u'\right) \quad (3)$$

If the equation is true, then the signature is valid. Otherwise, the signature is invalid.

3. Safety Integrity Verification Scheme Based on Schnorr Signature

We propose a verification scheme based on Schnorr Signature Scheme, which is named safety integrity verification scheme (SIVS). The parameters of SSIV are defined next. We suppose that p is 1024-bit prime; q is 160-bit prime, and $p-1$ is a multiple of q; g is a generator of Z_p^*, and $g^q = 1 \bmod p$; x is a private key of Z_q^*; y is a public key and $y = g^x \bmod p$; $h(\cdot)$ is an approved cryptographic hash function; $f(\cdot)$ is a pseudo-random function; ¤$(\cdot)$ is a pseudo-random permutation; $k_1, k_2, k_3 \leftarrow \{0,1\}^k$ are three keys, where k is the length of the three keys.

3.1. Pro-processing Phase

Before the user sends his the file F to the cloud storage server, he firstly splits file F into n blocks: $F=(m_1,m_2,\cdots,m_n)$, and each block has l bits. Then he uses pseudo-random function $f(\cdot)$ keyed with k_1 to derive random sequence $\{r_i\}_{1\le i\le n}$

$$r_i = f_{k_1}(r+i),\ (1\le i\le n) \tag{4}$$

Where r is an initial secret random number. To each file block m_i and each random number r_i $(1\le i\le n)$, the user computes:

$$u_i = g^{r_i}(\bmod p),\ s_i = xm_i + (\bmod q),\ (i\le i\le n),\ E = h(m_1 \| m_2 \| \cdots \| m_n) \tag{5}$$

The user sends the set of file blocks $F=(m_1,m_2,\ldots,m_n)$ and the set of verification blocks $U=(u_1,u_2,\ldots,u_n)$ to the cloud storage server and deletes their copies from his local storage. The user stores the set of signatures $S=(s_1,s_2,\ldots,s_n)$ and hash value E on the local, and they will be used on the verification and retrieve file phase.

3.2. Challenge Phase

Which storage device that the file F has been stored at? The user does not know, so he challenges the cloud storage server to verify whether the file is preserved intact in cloud or not. The user's challenge values are $Chal(fID,c,k_2,k_3,y)$, where fID is identity number of the file F; c is the number of challenged blocks, $1\le c\le n$; k_2 and k_3 are chosen randomly for each challenge; y is the user's public key.

3.3. Response Phase

After the cloud storage server has received the challenge values $Chal(fID,c,k_2,k_3,y)$, he uses pseudo-random permutation $¤(\cdot)$ keyed with k_2 to generate indices of challenged blocks $i_j = ¨_{k_2}(j)$ $(1\le j\le c,\ 1\le i_j\le n)$. Also, he uses pseudo-random function $f(\cdot)$ keyed with k_3 to derive coefficients $~_j = f_{k_3}(j)$ $(1\le j\le c, ~_j\in Z_q^*)$. Here, $~_1,~_2,\ldots,~_c$ are randomly generated for each challenge.

In pro-processing phase, the cloud storage server holds the set of file blocks $F=(m_1,m_2,\ldots,m_n)$ and the set of verification blocks $U=(u_1,u_2,\ldots,u_n)$. Grounded on the block indices $i_j (1\le j\le c)$, he chooses the subset of file blocks $\tilde{F}=(m_{i_1},m_{i_2},\ldots,m_{i_c})$ and the subset of verification blocks $\tilde{U}=(u_{i_1},u_{i_2},\ldots,u_{i_c})$, then computes:

$$± = \prod_{j=1}^{c}(u_{i_j})^{~_j}(\bmod p),\ v = \sum_{j=1}^{c} ~_j m_{i_j}(\bmod q),\ T = y^{-v}(\bmod p) \tag{6}$$

The cloud storage server sends response values $(T,±)$ to the user, and takes them as the proofs of possessing file F.

3.4. Verification Phase

After the user has received response values $(T,±)$, he also computes indices of challenged blocks $i_j = ¨_{k_2}(j)$ and random coefficients $~_j = f_{k_3}(j)$ $(1\le j\le c)$. Then the user chooses the subset of the signatures $\tilde{S}=(s_{i_1},s_{i_2},\ldots,s_{i_c})$ from the set of signatures $S=(s_1,s_2,\ldots,s_n)$ which has been saved previously on the local. Further, the user computes:

$$w = \sum_{j=1}^{c} \sim_j s_{i_j} (\mathrm{mod}\, q) \tag{7}$$

Then he checks the following equation:

$$\pm \stackrel{?}{=} g^w T (\mathrm{mod}\, p) \tag{8}$$

If the equation is true, then the user believes that the cloud storage server preserves well his file F. Otherwise, verification fails. The above equation holds because:

$$g^w T (\mathrm{mod}\, p) = g^w y^{-v} (\mathrm{mod}\, p) = g^{\sum_{j=1}^{c} \sim_j s_{i_j}} y^{-\sum_{j=1}^{c} \sim_j m_{i_j}} (\mathrm{mod}\, p)$$
$$= g^{\sum_{j=1}^{c} \sim_j (x m_{i_j} + r_{i_j})} g^{-x \sum_{j=1}^{c} \sim_j m_{i_j}} (\mathrm{mod}\, p) = g^{\sum_{j=1}^{c} \sim_j r_{i_j}} (\mathrm{mod}\, p)$$
$$= \prod_{j=1}^{c} g^{\sim_j r_{i_j}} (\mathrm{mod}\, p)$$
$$= \prod_{j=1}^{c} u_{i_j}^{\sim_j} (\mathrm{mod}\, p)$$
$$= \pm$$

3.5. Retrieve File Phase

At a later time, when the user needs his file F, he sends a request message req(fID) to the cloud storage server. After the cloud storage server receives message req(fID), he sends back file blocks $F' = (m_1', m_2', \ldots, m_n')$ to the user. The user uses hash function to compute:

$$E' = h\left(m_1' \| m_2' \| \cdots \| m_n'\right) \tag{9}$$

The user compares the set of hash values E' with E, and E has been saved on the local by himself in pro-processing phase. If $E' = E$, then $F' = F$, it means that the file blocks are intact. If $E' \neq E$, then $F' \neq F$, it means that some file blocks have been altered in network transmitting or on cloud storage [10].

4. Efficient Integrity Verification Scheme Based on Schnorr Signature

We modify SIVS scheme to attain a more efficient scheme based on Schnorr Signature Scheme, which is named efficient integrity verification scheme (EIVS), but it has weaker security guarantee than SIVS.

In pro-processing, the user computes:

$$u_i = g^{r_i} (\mathrm{mod}\, p),\ e_i = h(m_i \| u_i),\ s_i = x e_i + r_i (\mathrm{mod}\, q),\ (1 \le i \le n),\ E = e_1 \| e_2 \| \cdots \| e_n \tag{10}$$

Here, EIVS doesn't use file blocks $m_i\ (1 \le i \le n)$ to compute signatures $s_i\ (1 \le i \le n)$, but uses hash values $e_i\ (1 \le i \le n)$ to compute signatures. In response phase, all values of $\sim_j\ (1 \le j \le c)$ are set to 1, here, the cloud storage server computes:

$$\pm = \prod_{j=1}^{c} u_{i_j} (\mathrm{mod}\, p),\ e_{i_j} = h(m_{i_j} \| u_{i_j}),\ v = \sum_{j=1}^{c} e_{i_j} (\mathrm{mod}\, q) \tag{11}$$

Now, the scheme doesn't add all file blocks $m_i\ (1 \le i \le n)$ to generate v, but add all hash values $e_i\ (1 \le i \le n)$ to generate it.

In verification phase, all values of $\sim_j (1 \le j \le c)$ are also set to 1, then the user computes:

$$w = \sum_{j=1}^{c} s_{i_j} (\mathrm{mod}\, q) \tag{12}$$

Here, the user checks if below equation holds:

$$\pm \stackrel{?}{=} g^w T (\mathrm{mod}\, p) \tag{13}$$

5. Security and Performance Analysis

EIVS scheme substitutes hash values e_i for file blocks m_i to generate signatures s_i and v, also, all values of coefficients $\sim_j (1 \le j \le c)$ are set to 1, so it improves operation speed and reduces computational costs. But EIVS scheme can only verify the cloud storage server stores well the sum of hash values, and cannot guarantee that the cloud storage server preserves intact all file blocks. Therefore, in the short term, the user could use EIVS scheme to verify the integrity of his own file. In the long term, the user must use SIVS scheme to verify the integrity of the file to enhance the verification effect. To security analysis, we are only aimed at SIVS scheme. To performance analysis, we consider both schemes.

5.1. Security Analysis

The security of Schnorr Signature Scheme is based on the intractability of discrete logarithm problem, and the scheme satisfies the security notions in the random oracle model. The scheme has a shorter length of signature than RSA and ElGamal signature scheme at the same level of security. Our SIVS scheme are proposed based on Schnorr Signature Scheme, so it also satisfy the security notions in the random oracle model.

SIVS scheme gives double integrity verification guarantee to cloud storage data. One guarantee, in response and verification phase, the user checks response values $(T, \pm)$ to judge whether all file blocks are preserved intact in the cloud storage server. The other guarantee, in retrieve file phase, the user compares hash values E' with E to judge whether some file blocks have been altered in network transmitting or on cloud storage.

In response and verification phase, let us assume that the cloud storage server has lost some of file blocks, but preserves well all verification blocks, it can be proved that the cloud storage server can't pass through the user's integrity verification. The processes of this proof are as follows:

If the user and the cloud storage server chooses the subset of file blocks $\tilde{F} = (m_{i_1}, m_{i_2}, \ldots, m_{i_c})$ as challenged blocks, but the cloud storage server has lost file blocks $(m_{i_j}, \ldots, m_{i_k})$, where, $\{i_j, \ldots, i_k\} \subseteq \{i_1, \ldots, i_c\}$. Accordingly, the cloud storage server falsifies file blocks $(b_{i_j}, \ldots, b_{i_k})$ with $(m_{i_j}, \ldots, m_{i_k})$ replacement, then he computes:

$$v' = \sim_1 m_{i_1} + \ldots + \sim_j b_{i_j} + \ldots + \sim_k b_{i_k} + \ldots + \sim_c m_{i_c} (\mathrm{mod}\, p),\ T' = y^{-v'} (\mathrm{mod}\, p) \tag{14}$$

According to our hypothesis, the verification blocks $U = (u_1, u_2, \ldots, u_n)$ are stored perfectly in cloud. So when the cloud storage server chooses challenged verification blocks $\tilde{U} = (u_{i_1}, u_{i_2}, \ldots, u_{i_c})$ from U to generate $\pm$, the value of $\pm$ is no change with fake challenged file blocks.

After the user has received response values $(T', \pm)$, he computes the value of w, and verifies the relation $\pm \stackrel{?}{=} g^w T' (\mathrm{mod}\, p)$ whether is true or not. If the relation is true, then $T = T' (\mathrm{mod}\, p)$, this means $y^{-(\sim_1 m_{i_1} + \sim_2 m_{i_2} + \ldots + \sim_c m_{i_c})} = y^{-(\sim_1 m_{i_1} + \ldots + \sim_j b_{i_j} + \ldots + \sim_k b_{i_k} + \ldots + \sim_c m_{i_c})} (\mathrm{mod}\, p)$. If we could find out

$A \neq B \in Z_q^*$ to let $y^{-A} = y^{-B} (\mathrm{mod}\ p)$, then $y^{-v} = y^{-v'} (\mathrm{mod}\ p)$, but this is impossible[11]. So in verification phase, when the cloud storage server substitutes fake file blocks for original file blocks, he can't succeed on the user's integrity verification.

In retrieve file phase, we suppose the cloud storage server has lost original file blocks $m_l,\ldots,m_t$ When the cloud storage server substitutes fake file blocks $b_l,\ldots,b_t$ for file blocks $m_l,\ldots,m_t$ and send them to the user, the user first computes hash value $E' = h(m_1 \|,\ldots,\| b_l \|,\ldots,\| b_t \|,\ldots,\| m_n)$. Then he takes out $E = h(m_1 \| m_2 \| \cdots \| m_n)$ on the local, and checks below equation whether is true or not.

$h(m_1 \|,\ldots,\| b_l \|,\ldots,\| b_t \|,\ldots,\| m_n) \stackrel{?}{=} h(m_1 \| m_2 \|,\ldots,\| m_n)$

To make the equation true, unless the cloud storage server can find out the value of hash collision. This means he can find out hash values $h(A)$ and $h(B)$, let $h(A) = h(B)$ on the premise $A \neq B$, but this is not feasible [11]. In view of this, the user thinks that some file blocks have been altered in network transmitting or on cloud storage.

5.2. Performance Analysis

Comparing our SIVS and EIVS schemes with S-PDP in Ateniese [4] and Wang [13] schemes, in order to maintain the fairness, we don't consider communication and computation costs of root nodes and auxiliary authentication information in Wang [13] scheme. In the four schemes, if we suppose the size of each file block $|m_i|$ is the same, and total number of file blocks n is also the same. Moreover, the number of challenged file blocks c is also the same.

In the four schemes, communication costs are mainly composed of the costs of challenge and response values. In Wang [13] scheme, the TPA takes values $chal\{i, v_i\}$ as challenge values and sends them to the server. Moreover, the server returns the set of information $\{\text{«}, \pm, H(m_i), \Omega_i\}$ as response values to the TPA, so communication costs of Wang [13] scheme are the highest in the four schemes. However, the communication costs of SIVS, EIVS and S-PDP [4] schemes are roughly equivalent.

To computation costs, we ignore the costs that the server and the user derive challenge blocks indices i_j and random coefficient $\sim_j$ in the four schemes. We consider computation costs in pro-processing phase, in generating response values phase and in verifying response values phase. The computation costs of the three phases of the four schemes are listed in Table 1. In Table 1, the operation symbols denote meaning: Hash: hash function operation; Add: addition operation; Mult: multiplication operation; Exp: exponentiation operation; Div: division operation; Pair: pairing operation.

Table 1. Computational Costs Comparison of Four Schemes

Computational costs	S-PDP[4]	Wang[13]	SIVS	EIVS
pro-processing	2nExp+nMult +nHash	2nExp+nMult +nHash	nExp+nMult +1Hash+nAdd	nExp+nMult +(n+1)Hash+nAdd
generating response values	(c+1)Exp+2cMult +cAdd+1Hash	cExp+2cMult +cAdd	(c+1)Exp +2cMult +cAdd	1Exp+cMult+cHash +cAdd
verifying response values	(c+2)Exp+ cMult +1Div +(c+1)Hash	(c+1)Exp+(c+1)Mult +cHash+2Pair	1Exp+(c+1)Mult +cAdd	1Exp+1Mult +cAdd

Table 1 indicates computation costs of S-PDP [4] and Wang [13] schemes are roughly equivalent, and the computation costs of our SIVS and EIVS schemes are all lower than two other schemes. Moreover, the computation costs of EIVS scheme are the lowest in four schemes. To all operation, bilinear pairing and exponentiation operation are more time-consuming than hash and add operation. EIVS scheme has more hash operation than three other schemes, but the total exponentiation operation of EIVS scheme are less than three other schemes. Therefore, EIVS scheme is more efficient than three other schemes. Now we combine SIVS with EIVS scheme to check the integrity of cloud storage data. If the files are stored in cloud for a long time, we uses SIVS scheme. But in short term, we uses EIVS scheme.

Therefore, at the same level of security, our schemes in overall performance are superior to two other schemes and get double-efficient integrity verification guarantee.

6. Conclusion

In view of communication costs and computation costs of current integrity verification schemes are too high, this paper proposes two integrity verification schemes SIVS and EIVS based on Schnorr Signature. It combines SIVS with EIVS scheme to check the integrity of cloud storage data. If the files need to be stored in cloud for a long time, SIVS scheme will be used to verify the integrity of the file. But in short term, EIVS scheme will be used. Compared with other similar schemes, our schemes has lower computation costs and communication costs and get double-efficient integrity verification guarantee. How to apply the data recovery and privacy protection technology to improve fault tolerance and security of cloud storage data? It will become our emphasis in further research.

Acknowledgements

This work is partially surpported by Visiting Scholarship of State Key Laboratory of Power Transmission Equipment & System Security and New Technology (Chongqing University) under grant No. 2007DA10512711412.

References

[1] P Mell, T Grance. Reports on Computer Systems Technology. *The NIST Definition of Cloud Computing*. 2011.
[2] Cloud storage, Wikipedia, http://en.wikipedia.org/wiki/Cloud_storage
[3] Y Deswarte, JJ Quisquater, A Sadane. *Remote integrity checking*. 6th working conference on integrity and internal control in information systems (IICIS). 2003; 1-11.
[4] G Ateniese, R Burns, R Curtmola, et al. *Provable data possession at untrusted stores.* Proceedings of the 14th ACM conference on computer and communications security. New York, USA. ACM 2007; 598-609.
[5] R Curtmola, O Khan, R Burns, et al. *MR-PDP: Multiple-replica Provable data possession*. 28th IEEE ICDCS. 2008; 411-420.
[6] LX Chen. A homomorphic hashing based Provable Data Possession. *Journal of Electronics & Information Technology*. 2011; 33(9): 2199-2204.
[7] Juels A, Kaliski, BS Kaliski. *PORs: Proofs of Retrievability for large files.* Proceedings of the 14th ACM conference on Computer and communications security. ACM. 2007; 584-597.
[8] H Shacham, B Waters. *Compact proofs of retrievability*. Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security. 2008; 90-107.
[9] CP Schnorr. Method for identifying subscribers and for generating and verifying electronic signature in a data exchange system. U.S. 1991; Patent # 4995082.
[10] Song XL, Deng HY, et al. An Efficient Encryption and Verification Scheme for Preserving Electronic Evidence in Cloud Computing. *Journal of Information & Computational Science*. 2013; 3: 911-922.
[11] Liu FF, Gu D, Lu HN, et al. Reducing computational and communication complexity for dynamic provable data possession. *China Communications*. 2011; 6: 67-75
[12] Wang C, Wang Q, Ren K, et al. *Ensuring data storage security in cloud computing.* Proceedings of IWQos'09. 2009: 1-9.
[13] Q Wang, C Wang, K Ren, et al. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems*. 2011; 22(5): 847-859.