Cooperative Integrity Verification Schemes for Cloud Storage Data

Kishor Kumar Reddy C, Anisha P R
Vardhaman College of Engineering
India
kishoar23@gmail.com, anisha1990jan@gmail.com

ABSTRACT: *This paper proposes two integrity verification schemes based on Schnorr Signature Scheme, which are named Safety Integrity Verification Scheme (SIVS) and Efficient Integrity Verification Scheme (EIVS). In the two verification schemes, for the user's each challenge, the cloud storage server chooses randomly the sets of file blocks and verification blocks to generate response values, and the user uses the set of signatures to verify response values. That is to check whether the cloud storage server preserves perfectly the user's file or not. SIVS gives double integrity verification guarantee to cloud storage data. EIVS has more efficient verification guarantee in computational costs than SIVS. According to the different needs of users, EIVS cooperates with SIVS to improve efficiency when checking the integrity of cloud storage data. Compared with other schemes, at the same level of security, the two cooperative schemes get better integrity verification guarantee, and pay lower computational costs.*

Keywords: Cloud storage, Schnorr Signature Scheme, Double integrity verification, Challenge and response

Received: 23 December 2015, Revised 28 January 2016, Accepted 5 February 2016

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

nother 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 verifier challenge the server without limit, and the server converged the tags of all data blocks into a short tag that would be took 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*". As the server selected randomly some of data blocks to achieve the probability of successful verification, scheme [4] and [8] remedy partially some drawback of scheme [3] and [7]. 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.

p and q are two big prime, $\mathbf{p}$ - 1 and 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 (.) is an approved cryptographic hash function.

When the signer signs the message m, he chooses randomly a secret number $r \in Z_q^*$ and computes.

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

The signer sends the message m and the signatures (e, s) to the receiver.

When the receiver has received m and (e, s), in order to verify the validity of the signature, he first computes

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

Then he checks the following equation

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

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

3. Two Cooperative Integrity Verification Schemes

We propose two verification schemes based on Schnorr Signature Scheme, which are named safety integrity verification scheme (SIVS) and Efficient Integrity Verification Scheme (EIVS). The parameters of the two schemes are given the same definition, and they are phrased blow.

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(.) is an approved cryptographic hash function; f (.) is a pseudo-random function; φ (.) 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 Safety Integrity Verification Scheme (SIVS)

Safety integrity verification scheme (SIVS) and Efficient Integrity Verification Scheme (EIVS) both consist of five phases, such

that Pro-processing phase, Challenge phase, Response phase, Verification phase, and Retrieve File phase. The five phases of SIVS scheme are listed in Figure 1.

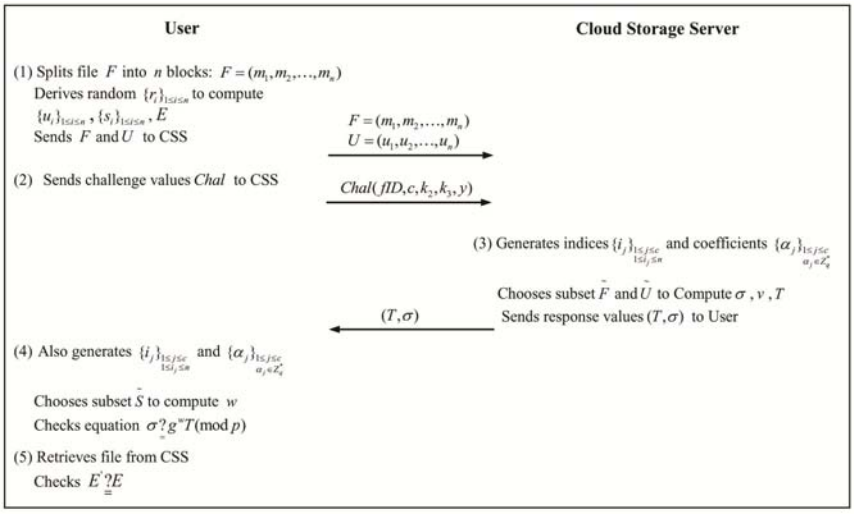

Figure 1.The proposed SIVS scheme

(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: $F = (m_1, m_2, ..., m_n)$, and the size of each block is represented as $|m_i|$ bits. Then he uses pseudo-random function $f(.)$ 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

$$u_i = g^{r_i} (\mathrm{mod}\, p), s_i = x m_i + r_i (\mathrm{mod}\, q) \ (1 \le i \le n) \tag{5}$$

$$E = h(m_1 \| m_2 \|, ..., \| m_n) \tag{6}$$

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

(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) 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 φ (.) keyed with k_2 to generate indices of challenged blocks $i_j = \varphi_{k_2}(j) \ (1 \le j \le c, 1 \le i_j \le n)$. Also, he uses pseudo-random function f (.) keyed with k_3 to derive coefficients $\alpha_j = f_{k_3}(j) \ (1 \le j \le c, \ \alpha_j \in Z_q^*)$. Here, $a_1, a_2, ..., a_c$ are randomly generated for each challenge

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

and the subset of verification blocks $\tilde{U} = (u_{i_1}, u_{i_2}, \ldots, u_{i_c})$, then computes:

$$\sigma = \prod_{j=1}^{c} (u_{i_j})^{\alpha_j} \pmod p \,, v = \sum_{j=1}^{c} \alpha_j m_{i_j} \pmod q \,, T = y^{-v} \pmod p \qquad (7)$$

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

(4) Verification Phase

After the user has received response values (T, σ) , he also computes indices of challenged blocks $i_j = \varphi_{k_2}(j)$ and random coefficients $\alpha_j = f_{k_3}(j)$ $(1 \le j \le c)$. 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:

$$w = \sum_{j=1}^{c} \alpha_j s_{i_j} \pmod q \qquad (8)$$

Then he checks the following equation

$$\sigma \stackrel{?}{=} g^{w} T \pmod p \qquad (9)$$

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:

$$\begin{aligned}
& g^{w} T \pmod p \\
&= g^{w} y^{-v} \pmod p \\
&= g^{\sum_{j=1}^{c} \alpha_j s_{i_j}} y^{-\sum_{j=1}^{c} \alpha_j m_{i_j}} \pmod p \\
&= g^{\sum_{j=1}^{c} \alpha_j (x m_{i_j} + r_{i_j})} g^{-x \sum_{j=1}^{c} \alpha_j m_{i_j}} \pmod p \\
&= g^{\sum_{j=1}^{c} \alpha_j r_{i_j}} \pmod p \\
&= \prod_{j=1}^{c} g^{\alpha_j r_{i_j}} \pmod p \\
&= \prod_{j=1}^{c} u_{i_j}^{\alpha_j} \pmod p \\
&= \sigma
\end{aligned}$$

(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

$$E' = h(m_1' \| m_2' \|, \ldots, \| m_n') \qquad (10)$$

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].

3.2 Efficient Integrity Verification Scheme (EIVS)

We simplify SIVS to attain a more efficient scheme based on Schnorr Signature Scheme, this is efficient integrity verification

scheme (EIVS). EIVS has the same parameters as SIVS, and it also includes five phases, but each phase of EIVS is simpler than that of SIVS. Here, only three main phases are described blow.

In pro-processing phase, the user computes

$$u_i = g^{r_i} (\mathrm{mod}\ p), e_i = h(m_i \| u_i), s_i = xe_i + r_i (\mathrm{mod}\ q)\ (1 \le i \le n) \tag{11}$$

$$E = e_1 \| e_2 \|, \ldots, \| e_n \tag{12}$$

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 a_j ($1 \le i \le c$) are set to 1, here, the cloud storage server computes

$$\sigma = \prod_{j=1}^{c} u_{i_j} (\mathrm{mod}\ p)\ ,\ e_{i_j} = h(m_{i_j} \| u_{i_j})\ ,\ v = \sum_{j=1}^{c} e_{i_j} (\mathrm{mod}\ q) \tag{13}$$

Now, EIVS 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 $a_j\ (1 \le i \le c)$ are also set to 1, then the user computes

$$w = \sum_{j=1}^{c} s_{i_j} (\mathrm{mod}\ q) \tag{14}$$

Here, the user checks if below equation holds

$$\sigma \stackrel{?}{=} g^{w} T (\mathrm{mod}\ p) \tag{15}$$

3.3 Two Schemes Cooperate for Cloud Storage Data

To play its due role, cloud storage system must be able to provide a good service for all users. The users of cloud storage are grouped into two classes: general public users and business users. As long as these users connect to Internet, they may have different requirements of cloud storage. Some users temporarily store their data in the cloud server, after a period of time they want the server to delete their data. Some users want to permanently store data in the cloud server. When they need these data, these data can be retrieved from the server at any time. According to the different requirements of users, the two schemes SIVS and EIVS will cooperate to check the integrity of users' data in cloud.

In 3.2, EIVS substitutes hash values e_i for file blocks m_i to generate signatures s_i and v, also, all values of coefficients a_j ($1 \le i \le c$) are set to 1, so it improves operation speed and reduces computational costs. But EIVS can only verify that the cloud storage server stores well the sum of hash values, and cannot ensure that the cloud storage server preserves intact all file blocks. From 3.1, SIVS provides double integrity verification guarantee to cloud storage data. There will be a detailed description on security guarantee of SIVS in 4 chapter. Therefore, in general, EIVS should be adequate for cloud data stored temporarily. SIVS should be adequate for cloud data that need to be stored permanently and do not tolerate any loss and damage. So EIVS should cooperate with SIVS to improve efficiency when checking the integrity of cloud storage data.

4. Security and Performance Analysis

For cloud storage data, SIVS scheme provides safer guarantee of integrity verification. EIVS is a simplified version of SIVS, and it is more efficient in computational costs than SIVS but with weaker security guarantee. So following security analysis, we are only aimed at SIVS scheme and ignore EIVS. To performance analysis, we consider both schemes.

4.1 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. SIVS is proposed based on Schnorr Signature Scheme, so it also satisfy the security notions in the random oracle model.

SIVS gives double integrity verification guarantee to cloud storage data. One guarantee, in response and verification phase, the user checks response values (T, σ) 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 choose 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

$$v' = \alpha_1 m_{i_1} + \ldots + \alpha_j b_{i_j} + \ldots + \alpha_k b_{i_k} + \ldots + \alpha_c m_{i_c} \pmod p \tag{16}$$

$$T' = y^{-v'} \pmod p \tag{17}$$

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 σ, the value of σ is no change with fake challenged file blocks.

After the user has received response values (T', σ), he computes the value of w, and verifies the relation $\sigma \stackrel{?}{=} g^{w} T' \pmod p$ whether is true or not. If the relation is true, then $T = T' \pmod p$, this means $y^{-(\alpha_1 m_{i_1} + \alpha_2 m_{i_2} + \ldots + \alpha_c m_{i_c})} = y^{-(\alpha_1 m_{i_1} + \ldots + \alpha_j b_{i_j} + \ldots + \alpha_k b_{i_k} + \ldots + \alpha_c m_{i_c})} \pmod p$. If we could find out $A \neq B \in Z_q^*$ to let $y^{-A} = y^{-B} (mod\ p)$, then $y^{-v} = y^{-v} (mod\ p)$, but this is impossible. 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 $E = h(m_1 \| m_2 \|, \ldots, \| m_n)$ 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.

4.2 Performance Analysis

Comparing 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. Also, we don't consider the computation costs that the user and the cloud storage server derive random sequence $\{r_i\}_{1 \le i \le n}\}$, challenge blocks indices $\{i_j\}_{\substack{1 \le j \le c \\ 1 \le i_j \le n}}$ and random coefficients $\{\alpha_j\}_{\substack{1 \le j \le c \\ \alpha_j \in Z_q^*}}$ in the four schemes.

When we analyze the performance of the four schemes, 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 $\{\mu, \sigma, H(m_i), \Omega_i\}$ 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 of challenge phase and retrieve file phase, and only consider the costs of pro-processing phase, response phase and verification operation; *Add*: addition operation; *Mult*: multiplication operation;

Computational costs	S-PDP[4]	Wang[13]	SIVS	EIVS
Pro-processing phase	2nExp+nMult +nHash	2nExp+nMult +nHash	nExp+nMult +1Hash+nAdd	nExp+nMult +(n+1)Hash+nAdd
Response phase	(c+1)Exp+2cMult +cAdd+1Hash	cExp+2cMult +cAdd	(c+1)Exp +2cMult +cAdd	1Exp+cMult+cHash +cAdd
Verification phase	(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. Computational Costs Comparison of Four Schemes

Table 1 indicates computation costs of S-PDP [4] and Wang[13] schemes are roughly equivalent, and the computation costs of SIVS and EIVS schemes are all lower than two other schemes. To all operation, bilinear pairing operation is most time-consuming, followed by exponentiation operation. Multiplication operation is also more time-consuming than hash and add operation. EIVS has more hash operation than three other schemes, but has no pairing operation. Moreover the total exponentiation operation of EIVS are less than three other schemes. So the computation costs of EIVS are the lowest in four schemes, and the costs of SIVS are next lowest. Therefore, at the same level of security, when SIVS and EIVS cooperate to check the integrity of cloud storage data, we can get more efficient integrity verification guarantee than other schemes.

5. 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. SIVS and EIVS cooperate 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 ensure double integrity of cloud data. But for cloud data stored temporarily, EIVS will be used to get 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.

Acknowledgment

This work is partially supported by Program for Innovation Team Building at Institutions of Higher Education in Chongqing under Grant No. KJTD201310, Natural Science Foundation of Chongqing Science &Technology Commission of China under Grant No. 2011jjA40031, Science &Technology Research Foundation of Education Committee of Chongqing of China under Grant No. KJ120504.

References

[1] Mell, P., Grance, T. (2011). The NIST Definition of Cloud Computing, Special Publication 800–145. National Institute of Standards and Technology: Gaithersburg, MD, USA.

[2] Wikipedia, Cloud storage. (2007). http://en.wikipedia.org/wiki/Cloud_storage (accessed onJuly 2014).

[3] Deswarte, Y., Quisquater, J.-J. (2003). A.Sadane, Remote integrity checking. *In*: 6th working conference on integrity and internal control in information systems (IICIS), 1-11.

[4] Ateniese, G., Burns, R., Curtmola, R., et al (2007). Provable data possession at untrusted stores. *In*: Proceedings of the 14th ACM conference on computer and communications security, p. 598-609.

[5] Curtmola, R., Khan, O., Burns, R., et al (2008). MR-PDP: Multiple-replica provable data possession. *In*: 28th IEEE ICDCS, 411-420.

[6] Chen, L.X. (2011). A homomorphic hashing based Provable Data Possession. *Journal of Electronics & Information Technology* 33 (9) 2199-2204.

[7] Juels, A., Kaliski, B. S. (2007). PORs: Proofs of Retrievability for large files. *In*: Proceedings of the 14th ACM conference on Computer and communications security, 584-597.

[8] Shacham, H. Waters, B (2008). Compact proofs of retrievability. *In*: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, p. 90-107.

[9] Schnorr, C.P (1991). Method for identifying subscribers and for generating and verifying electronic signature in a data exchange system, U.S. Patent # 4995082.

[10] Zhu, Y., Hu, H.X., Ahn, G. J., Yu, M.Y. (2012). Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Transactions on Parallel and Distributed Systems*, 23 (12) 2231 - 2244.

[11] Liu, F.F., Gu, D., Lu, H. N., et al (2011). Reducing computational and communication complexity for dynamic provable data possession, *China Communications*, 8 (6) 67-75.

[12] Wang, C., Wang, Q., Ren, K., et al. (2009). Ensuring data storage security in cloud computing. *In:* Proceedings of IWQos'09, p. 1-9.

[13] Wang, Q., Wang, C., Ren, K. (2011). Enabling public auditability and data dynamics for storage security in cloud computing, *IEEE Transactions on Parallel and Distributed Systems* 22 (5) 847-859.