Novel Data Integrity Verification Schemes in Cloud Storage

Thanh Cuong Nguyen, Wenfeng Shen, Zhaokai Luo, Zhou Lei and Weimin Xu

Abstract In order to persuade users of widely using cloud storage, one critical challenge that should be solved is finding way to determine whether data has been illegally modified on the cloud server or not. The topic has although been addressed in several works, there is lack of scheme to meet all the demand of supporting dynamic operations, public verification, less computation etc. This paper surveys the related results that has been done and proposes two alternative schemes, called DIV-I and DIV-II. Compared to S-PDP introduced by Ateniese et al., both DIV-I and DIV-II use less time to generate tags and verify. In addition, the proposed schemes fully support dynamic operations as well as public verification.

Keywords Cloud storage · Data integrity · Dynamic data

1 Introduction

When using cloud storage, user enjoys many prominent characteristics such as accessing data anytime and anywhere, being released from arduous work of maintaining hardware and software, etc. Those cutting edges promote the wide adaptation of cloud storage in practice. However, many critical security challenges emerge due to user's limited control over his data, which totally differentiate to traditional storage approach. One of those challenges is how to determine the intactness of data is ensured or not. Notice that the data can be damaged by many reasons from malicious attack to hardware failure, and the cloud storage providers (CSP) may hide that to hold their reputation. Hence, establishing a scheme to verify the data's intactness is a key requirement.

T.C. Nguyen (✉) · W. Shen · Z. Luo · Z. Lei · W. Xu
School of Computer Engineering and Science,
Shanghai University, Rm. 809, Computer Building, 333 Nanchen Road, Shanghai 200444, China
e-mail: ruan3@shu.edu.cn

W. Shen
e-mail: wfshen@mail.shu.edu.cn

R. Lee (ed.), *Computer and Information Science*, Studies in Computational Intelligence 566, DOI 10.1007/978-3-319-10509-3_9

Such a verification scheme should have the following features, as many as possible:

Support public verification: The scheme should allow not only data owner but also anyone authorized in the cloud system to verify the data integrity. This is especially important because when outsourcing data, user sometime wants to share his data to his friends or partners. They need to have ability to make sure that the data they retrieve are not illegally modified. Additionally, user, who is not online frequently, can delegate the verification task to a third party auditor (TPA). The TPA will send the verification request periodically to CSP to ensure that any data corruption is detected in time. TPA can even play more essential role in evaluating the quality of service of CSP. Any CSP that has poor profile in terms of ensuring the innocence of the data it stores will probably lose their users.

Unlimited verification time: For each verification request, auditor needs to send a challenge to CSP, and CSP is supposed to return appropriate result corresponding to the challenge. If the scheme allows only limited verification time, some challenges must be repeated after all of them has been used. That brings CSP a wonderful chance to answer with a deceived return if it stores all of previous response corresponding to each challenge.

No data leakage: During the verification process, the data owner may not want to reveal any data to TPA. Hence, the scheme should protect the data privacy against TPA no matter how many tuple (challenge, response) is collected.

Support data dynamic operations: The data owner sometime wants to modify his file such as deleting part of the file or inserting more data somewhere in the file. In this case, the file's tags need to be recomputed. Thus, the scheme should be able to update the tags with lowest cost.

This paper proposes two schemes that address all above concerns. The proposed schemes fully support data dynamic operations as well as public verification without data leakage. Related demonstration and experiment are carried out to prove their correctness and outstanding performance compared to other schemes.

The rest of this paper is organized as follows. Section 2 reviews some related work. Section 3 introduces two novel schemes DIV-I and DIV-II in detail. Section 4 presents experiment results and analysis. And the last section makes conclusion.

2 Related Work

Many outstanding work has been done to provide judgment scheme for data integrity on cloud storage. Some of them generates limited number of tuple (challenge, response) before outsourcing data. For example, Juels and Kaliski introduced POR scheme [1] which embedded a number of special blocks, called 'sentinels', among file blocks, and the verifier releases one sentinel's position for each challenge. Aravan and Ashutosh [2] selected from each block a certain number of bits to compose its verification proof. This scheme has been improved in [3] to support data dynamic operations, but limited verification time still remains. Ateniese et al. [4] used cryptographic hash

function to generate verification proof. This scheme supports block update, deletion and append. However, block insertion anywhere in the file is not allowed. Chaoling li et al. [5] tried to improve scheme of [4] to support block insertion by using SN-BN table. However, the authors did not realizes that correctness of this scheme was not ensured if the integrity of SN-BN table was not guaranteed. Noticeably, if the CSP stores full set of tuples, all above schemes are not reliable any more.

On the other hand, some solutions based on RSA and Diffie-Hellman assumptions support dynamic operations such as Deswarte and Quisquater [6] and Sebe et al. [7]. However, those schemes do not support public verification. Some another solutions using homomorphic authentication like Shacham and Waters [8] and Ateniese et al. [9] allow unlimited public verification time, but may lead to data leakage. We notice that S-PDP introduced in [9] can be improved by masking the CSP's response with a random number to prevent data leakage. Erway et al. [10] proposed another model based on Ateniese et al.'s [9] model to better support dynamic operations. Liu et al. [11] improved Erway et al.'s [10] model to reduce computational and communication overhead. Neither of them allows public verification. Wang et al. [12] took advantage of both homomorphic authentication and Merkle hash tree [13] to allow both unlimited public verification and dynamic operations. However, data privacy was not considered in this work. Zhu [14] introduced a scheme based on Diffie-Hellman assumption [15] and bilinear pairings. Although the scheme supports public verification, dynamic operations were not addressed in this work.

3 The Novel Schemes

Let p, q, be two large primes and $N = p * q$ be an RSA modulus. Let $\varphi(N) = (p-1) * (q-1)$ be the Euler function of N, and d, e are two big integers satisfy $d * e \equiv 1 \bmod \varphi(N)$. Let l is a security parameter, assume that $|d| \geq l$, $|e| \geq l$, ($|d|$, $|e|$ are bit-length of d and e respectively). N and e are made public while p, q, $\varphi(N)$, d are only known by the data owner. Additionally, let g be an element with high order in $\mathbb{Z}_N^*$ and g is coprime to N. g is also made publicly known.

We suppose that data owner has a file, which includes n blocks, each block has bits, needs to be outsourced. Hence, the file size is $n * s_b$ bits. In this paper, a tag is calculated for each block as its authentication data. The ith block is denoted by b_i and its tag is denoted by T_i.

3.1 DIV-I Scheme

Here, we propose a scheme which is similar to S-PDP but performances better in term of tag generation and integrity verification. We call the scheme DIV-I. The scheme includes four functions as follows:

GenKey$(l_1, l_2) \rightarrow (pk, sk)$: generates a public key $pk = (N, e, v, r, g)$ and private key $sk = (\varphi(N), d, \alpha)$, where r, α are two random numbers and $v = g^{\alpha} \; mod \; N$. Let l_1, l_2 are two security parameter, $r \leftarrow \{0, 1\}^{l_1}, \alpha \leftarrow \{0, 1\}^{l_2}$

GenTag$(pk, sk, i, b_i) \rightarrow T_i$: Let $h_i = H(r||i)$, where H is responsible to compute the hash value and convert to big integer. Data owner computes $T_i = g^{(\alpha h_i + b_i) * d} \; mod \; N$ and sends $\{(b_i, T_i)\}_{0 \leq i < n}$ to CSP.

GenProof$(pk, F, chal) \rightarrow (T, B)$: Verifier sends challenge to CSP in a form of $(g_s, c, \{(i_j, c_j)\}_{1 \leq j \leq c})$, where $g_s = g^s \; mod \; N$, s is a random number, c is the number of blocks that compose the response, (i_j, c_j) is those blocks' index and coefficient, correspondingly. CSP responds two values:

$$T = \Pi_{j=1}^{c} T_{i_j}^{c_j} \; mod \; N \text{ and } B = g_s^{\sum_{j=1}^{c} c_j b_{i_j}} \; mod \; N$$

VenProof$(pk, chal, T, B) \rightarrow \{''Y'', \; ''N''\}$: Verifier computes $h = v^{\sum_{j=1}^{c} c_j h_{i_j}} \; mod \; N$ and check the condition $T^{e*s} = B * h^s \; mod \; N$. If the condition is satisfied, then returns "Y" indicates no data corruption detected. Otherwise, returns "N" indicates Data corruption detected.

Correctness: In case of the intact of all b_{i_j} and T_{i_j} is ensured, we have $T^{e*s} = g^{s * \sum_{j=1}^{c} (c_j b_{i_j} + \alpha c_j h_{i_j})} \; mod \; N$

$$\begin{aligned} &= g^{s * \sum_{j=1}^{c} c_j b_{i_j}} * g^{\alpha s * \sum_{j=1}^{c} c_j h_{i_j}} \; mod \; N \\ &= B * (g^{\alpha * \sum_{j=1}^{c} c_j h_{i_j}})^s \; mod \; N \\ &= B * h^s \; mod \; N \end{aligned}$$

Robustness: Assuming that the factorization, RSA and Diffie-Hellman problem are difficult over $\mathbb{Z}_N^*$, CSP successfully pass the challenge for DIV-I scheme if and only if the intact of all blocks and tags participate in the response is ensured.

Proof Suppose that some of blocks participate in the response are corrupted. We prove that if CSP successfully pass the challenge, there is method to break RSA problem. That means with any integer z, it is able to find a value w that satisfies $w^e = z \; mod \; N$ without knowing d. The construction of this method is describes as follows.

In the above construction of DIV-I, let $g = z$. We assume that there are k corrupted blocks in total c blocks compose B, and w.l.o.g they are $b'_{i_1}, b'_{i_2}, \ldots, b'_{i_k}$. CPS responds to verifier a tuple (T', B'), where $B' = g_s^{b'} \; mod \; N$. Without knowing s, in order to pass the challenge, CSP needs to ensure that $T'^{e} = g^{b'} * h \; mod \; N$ (2). Let $u = \sum_{j=1}^{c}, c_j h_{i_j}$ (2) $\Leftrightarrow T'^{e} = z^{b'} * z^{\alpha u} \; mod \; N$. Because α is unknown to CSP and we can choose e as a large prime, thus w.l.o.g we assume that $gcd(e, b' + \alpha u) = 1$. Thus the extended Euclidian algorithm can be used to find out x and y such that $ex + (b' + \alpha u)y = 1$. Let $w = z^x * T'^{y}$, we have $w^e = z^{ex} * T'^{ey} = z^{ex} * z^{(b' + \alpha u)y} = z$. That means the RSA problem has been break.

Update: If data owner inserts a block b at the position pos, all the tag of blocks from pos to n need to be updated. The update process is summarized as follow:

- CSP sends $\{T_i\}_{pos \le i \le n}$ to data owner.
- Data owner computes new tags: $T'_{i+1} = T_i * g^{\alpha(h_{i+1}-h_i)*d} \, mod \, N$
- Data owner sends $\{T'_{i+1}\}_{pos \le i \le n}$ to CSP.

The maximum communication overhead is $(2n+1)*|N|+s_b$ when $pos = 0$ and the minimum one is $|N| + s_b$ when $pos = n + 1$ ($|N|$ is bit-length of N). The deletion process is carried out similarly.

Storage: pk and sk need a storage space of $5|N|+l_1+l_2$ bits (g is ignored because we can set g as small integer like 2, 3, 5). Extra storage to keep all tags on CSP is $*|N|$. Thus, the ratio of extra storage and file size is $|N|/s_b$. For example, in case of a 4GB file includes n =1,000,000 blocks, each block is 4 KB-size, $|N| = 1024$ bits and $l_1 = l_2 = 128$ bits, CSP need 122 MB extra storage, pk and sk need 5.25 KB.

3.2 DIV-II Scheme

We try to reduce power and multiplication computation compared to above schemes to hopefully lessen the computation cost. The new scheme, called DIV-II, is described as follow:

GenKey$(l_1, l_2) \rightarrow (pk, sk)$: generates a public key $pk = (N, e, r, g, v)$ and private key $sk = (\varphi(N), d, \alpha, u)$, where r, u are random numbers and $v = g^{\alpha} \, mod \, N$. Let l_1, l_2, are two security parameter, $r \leftarrow \{0, 1\}^{l_1}, u \leftarrow \{0, 1\}^{l_1}, \alpha \leftarrow \{0, 1\}^{l_2}$.

GenTag$(pk, sk, i, b_i) \rightarrow (T_i, \beta_i)$: Let $h_i = H_1(r||i)$, $h'_i = H_2(u||i)$ and $\beta_i = g^{h_{i'}} \, mod \, N$, where H_1, H_2 are two functions that compute the hash value of string and convert to big integer. Data owner generates tag for all block using the formula $T_i = \alpha * b_i + d * h_i + h'_i \, mod \, \varphi(N)$ mod then sends $\{(b_i, T_i, \beta_i)\}_{0 \le i \le n}$to CSP.

GenProof$(pk, F, chal) \rightarrow (T, B)$: Verifier sends challenge to CSP in a form of $(c, \{(i_j, c_j)\}_{1 \le j \le c})$, where is the number of blocks that compose the response, (i_j, c_j) is those blocks' index and coefficient, correspondingly. CSP responds three values:

$$T = g^{\sum_{j=1}^{c} c_j T_{i_j}} \, mod \, N, B = v^{\sum_{j=1}^{c} c_j b_{i_j}} * \Pi_{j=1}^{c} \beta_{i_j}^{c_j} \, mod \, N.$$

VerProof$(pk, chal, T, B) \rightarrow \{"Y", "N"\}$: Verifier computes $h = g^{\sum_{j=1}^{c} c_j h_{i_j}} \, mod \, N$ and check the condition $T^e = B^e * h \, mod \, N$. If the condition is satisfied, then return "Y" means "No data corruption detected". Otherwise, return "Y" means "Data corruption detected".

Correctness: In case of the intact of all b_{i_j}, T_{i_j} and β_{i_j} is ensured, we have: $T^e = g^{e*\sum_{j=1}^{c} c_j T_{i_j}} \, mod \, N$

$$= g^{e*\sum_{j=1}^{c} c_j(\alpha*b_{i_j}+d*h_{i_j}+h'_{i_j})} \, mod \, N$$
$$= g^{e*\alpha*\sum_{j=1}^{c} c_j b_{i_j}} * g^{\sum_{j=1}^{c} c_j h'_{i_j}} * g^{\sum_{j=1}^{c} c_j h_{i_j}} \, mod \, N$$
$$= B^{e*} h \, mod \, N$$

Robustness: Assuming that the factorization, RSA and Diffie-Hellman problem are difficult over $\mathbb{Z}_N^*$, CSP successfully pass the challenge for DIV-II scheme if and only if the intact of all blocks and tags participate in the response is ensured.

Proof We prove that if CSP's response to challenge can successfully pass the verification while some block has been corrupted, there is scheme to break RSA problem. For instance, for a particular z , we are able to find out a value w that satisfies $W^e = z \, mod \, N$.

In the above scheme, let $g = z$. Suppose that CSP responds T and B. Let $u = \sum_{j=1}^{c} c_j h_{i_j}$. If the response passes the verification, the equation $T^e = B^e * z^u \, mod \, N$ (3) should be established. (3)$\Leftrightarrow$ $(T B^{-1})^e = z^u \, mod \, N$. We assume w.l.o.g that $gcd(e, u) = 1$. Thus we can find out x and y such that $ex + uy = 1$. Let $w = z^x * (T B^{-1})^y$, we have $w^e = z^{ex} * (T B^{-1})^{ey} = z^{ex} * z^{uy} = z$. That means the RSA problem has been break.

Update: In DIV-II, when data owner inserts a block at the position , all the tags need to be updated. The update process is summarized as follow:

- CSP sends $\{T_i\}_{0 \le i \le n}$ to data owner.
- Data owner chooses a random number u'. Let $h''_i = H_2(u'||i)$
- Data owner computes new tags:

$$T'_i = T_i + h''_i - h'_i \, mod \, \varphi(N) \text{ if } 0 \le i < pos$$
$$T'_{i+1} = T_i + d * (h_{i+1} + h_i) + h''_{i+1} - h'_i \, mod \, \varphi(N) \text{ if } pos \le i < n$$

- Data owner sends $\{T'_i\}_{0 \le i \le n}$ to CSP.

Like the previous scheme, the update process of DIV-II is without downloading the block's data. The communication overhead is $(2n+1)*|N|+s_b$. However, if new block is appended, no tag need to be updated. Thus, the communication overhead is only $|N| + b_s$.

Unlike DIV-I scheme, when a new block inserted, DIV-II requires all tag to be recomputed. If tag of blocks prior position are not recomputed, there's no need to choose u'. Hence, $T'_{i+1} = \alpha * b_i + d * h_{i+1} + h'_{i+1} \, mod \, \varphi(N)$, $pos \le i < n$. Because CSP knows $T_{i+1} = \alpha * b_{i+1} + d * h_{i+1} + h'_{i+1} \, mod \, \varphi(N)$, $pos \le i < n$ it can compute $T'_{i+1} - T_{i+1} = \alpha * (b_i - b_{i+1}) \, mod \, \varphi(N)$. Thus, CSP can find out α as well as $\varphi(N)$, and it will be able to compute all the tag.

Storage: compared to DIV-I, the ratio of extra storage on CSP and file size is $2|N|/s_b$.

In both DIV-I and DIV-II, in order to avoid storing many r, we can set this value to each file's unique sequence provided by CSP. Additionally, if we use two seeds,

one used to generate i_j and the other used to generate c_j, just like S-PDP scheme, the challenge communication overhead can be reduced from $0(c)$ to $0(1)$. Besides that, unlike some previous schemes, verifier does not receive a linear combination of data blocks, hence data privacy is preserved.

The deletion process is similar to insertion.

4 Experiment and Analysis

4.1 Experiment Environment and Assumptions

We code C++ programs to run on a computer with Intel(R) Core(TM) 2 Quad CPU Q8200@2.33 GHz, 2.34 GHz and RAM of 2 GB to test the performances. We want to compare the computation time of two proposed schemes with S-PDP scheme, thus we just need one computer to run all functions including GenKey, GenTag, GenProof, VerProof. We do not use two seeds to generate i_j and c_j. Those values are included in the challenge. Additionally, we use MD5 to compute all hash value. Algorithms use NTL library [16] for doing big number calculation. Each case in our experiment is repeated 10 times and the mean value is used for further analysis.

4.2 Results and Analysis

First of all, we test the performance of three schemes (S-PDP, DIV-I, DIV-II) for 1 MB file in different cases corresponding to different value of block size. As can be seen from Fig. 1, the tag generation time of all three scheme is inversely proportional to block size. Additionally, the time of S-PDP is the biggest and the time of DIV-II is significantly less than that of the others. Interestingly, Fig. 2 shows that the server computation time is absolutely the same in three scheme independently to block size. Moreover, the time decreases when block size is smaller than 4 KB, reaches the bottom when block size is equal to 4 KB, and increases with larger block. Table 1 presents that the verification time of S-PDP is inversely proportional to block size while the time of other schemes just slightly decreases when block size varies from 1 to 64 KB. Furthermore, the time of our two schemes is less than that of S-PDP in all cases.

On the other hand, Fig. 3 depicts inversely proportional relation of maximum insertion time and block size for two novel schemes. Notably, the time of DIV-I is the bigger than that of DIV-I. In Fig. 4, however, the minimum insertion time of DIV-II is the bigger. In fact, the maximum and minimum insertion time of DIV-II is the same because all blocks need to be updated in both situations.

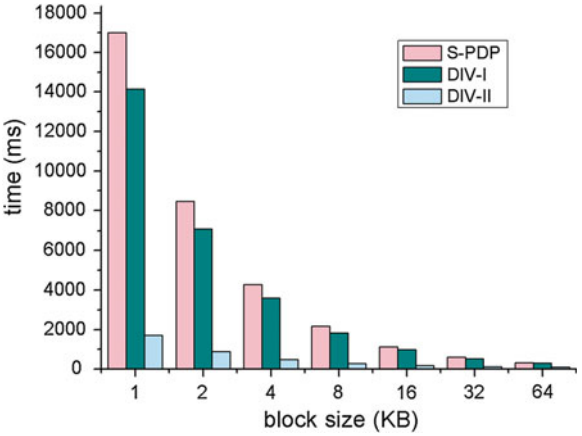

Fig. 1 Tag generation time for 1 MB file

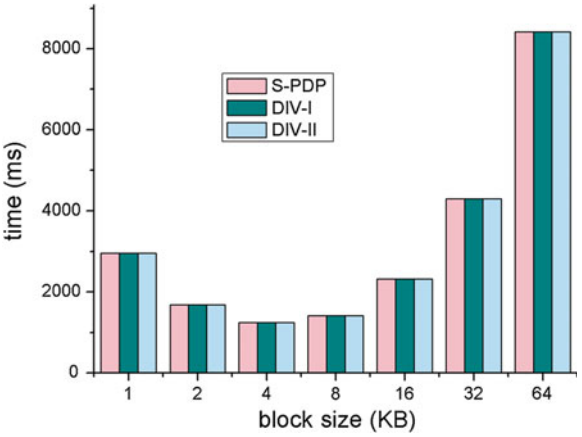

Fig. 2 Server computation time for 1 MB file

Table 1 Verification time for 1 MB file

Block size (KB)	S-PDP (ms)	DIV-I (ms)	DIV-II (ms)
1	2602	32	43
2	1298	27	39
4	659	27	37
8	337	25	37
16	180	25	37
32	97	24	36
64	58	24	36

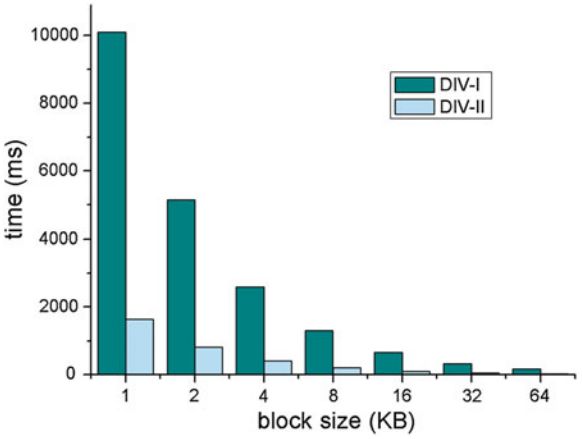

Fig. 3 Maximum insertion time for 1 MB file

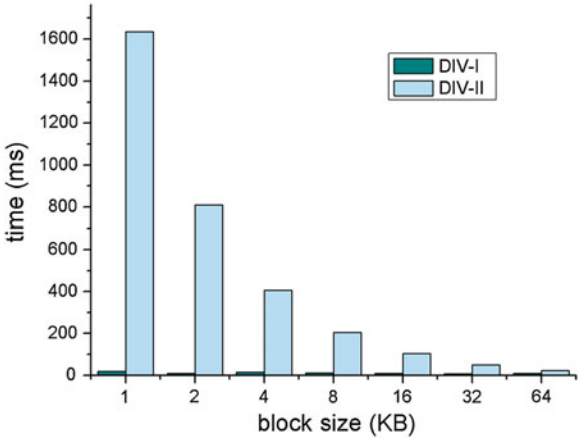

Fig. 4 Minimum insertion time for 1 MB file

Next, we set block size to 4 KB and test the performance of three scheme with different file size. Note that when block size is constant, number of block is directly proportional to file size. Figure 5 shows the directly proportional relation of server computation time, which is the same for three schemes, and file size. However, in case of verification, as can be seen from Table 2, the time of our two schemes only lightly rises while that of S-PDP still directly proportional to file size and the time of S-PDP is always bigger than that of two proposed schemes in all cases.

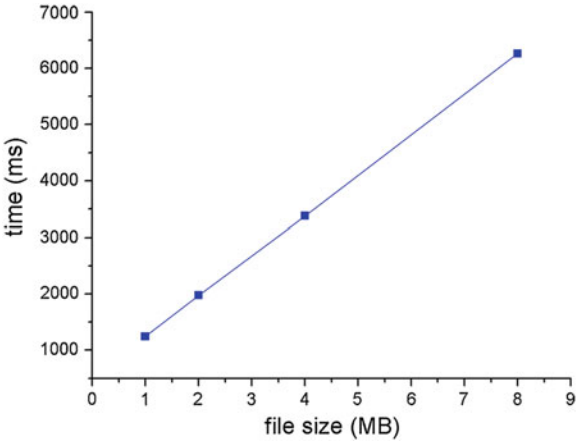

Fig. 5 Server computation time for 4 KB block size

Table 2 Verification time in case of 4 KB block size

File size (MB)	S-PDP (ms)	DIV-I (ms)	DIV-II (ms)
1	659	27	37
2	1311	28	40
4	2593	30	43
8	5160	37	48

5 Conclusion and Future Work

This paper proposes two novel schemes to verify the data integrity in cloud storage. Those schemes allow unlimited verification time and third party verification. Moreover, those support public verification and do not introduce any data leakages. Compare to S-PDP, those schemes need fewer computation time to generate tags and verify data integrity. Additionally, DIV-II needs more extra storage on CSP and dramatically decreases tag generation time compared to DIV-I. However, the verification time of DIV-II is slightly bigger than that of DIV-I. Interestingly, we notice that the two novel scheme have potential to combine with error-correcting like in POR and with spot checking referred in [9] to obtain better performance. This idea should be addressed in the future work.

Acknowledgments This study was supported by National High-tech R&D Program of China (Grant NO. 2009AA012201), the Shanghai Leading Academic Discipline Project (Project No.J50103), and the Innovation Project of Shanghai University.

References

1. Juels, A., Kaliski, B.S.Jr: PORs : proofs of retrievability for large files. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, pp. 584–597 (2007)
2. Sravan Kumar, R., Saxena, A.: Data integrity proofs in cloud storage. In: 2011 3rd International Conference on Communication Systems and Networks (COMSNETS), pp.1–4 (2011)
3. Nguyen, T.C., Shen, W., et al.: A probabilistic integrity checking scheme for dynamic data in untrusted cloud storage. In: 12th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2013), Niigata, Japan, pp. 179–183 (2013)
4. Ateniese, G., Pietro Di, R., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: Proceedings of the 4th iIternational Conference on Security and Privacy in Communication Networks - SecureComm 08, pp. 1–10 (2008)
5. Li, C., Chen, Y., Tan, P., Yang, G.: An efficient provable data possession scheme with data dynamics. International Conference on Computer Science & Service System (CSSS), pp. 706–710 (2012)
6. Deswarte, Y., Quisquater, J.: Remote integrity checking. IFIP Int. Fed. Inf. Process. **140**, 1–11 (2004)
7. Sebe, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y.: Efficient remote data possession checking in critical information infrastructures. IEEE Trans. Knowl. Data Eng. **20**(8), 1034–1038 (2008)
8. Shacham, H., Waters, B.: Compact proofs of retrievability. In: ASIACRYPT '08 Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, pp. 90–107 (2008)
9. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security - CCS '07, pp. 598–609 (2007)
10. Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: Proceedings of the 16th ACM conference on Computer and Communications Security - CCS '09, pp. 213–234 (2009)
11. Liu, F., Gu, D., Lu, H., Chris, C.: An improved dynamic provable data possession model. In: IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), pp. 290–295 (2011)
12. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of the 14th European Symposium on Research in Computer Security, ESORICS 2009, pp. 355–370 (2009)
13. Merkle, R.C.: Protocols for public key cryptosystems. In: Proceedings of IEEE Symposium on Security and Privacy'80, pp. 122–133 (1980)
14. Zhu, Y., Hu, H., Ahn, G., Yau, S.S.: Efficient audit service outsourcing for data integrity in clouds. J. Syst. Softw. **85**(5), 1083–1095 (2012)
15. Dan, B.: The decision Diffie-Hellman problem. In: Proceedings of the 3rd Algorithmic Number Theory Symposium. Lecture Notes in Computer Science 1423, pp. 48–63 (1998)
16. Shoup, V.: NTL: a library for doing number theory. http://www.shoup.net/ntl/. Accessed 29 Dec 2013