A new remote data integrity checking scheme for cloud storage

Yan Xiangtao[1*], Li Yifa[1]
[1] Information engineering university, Zhengzhou, China
taoexcellent@163.com

Abstract: Cloud storage services enable user to enjoy high-capacity and high-quality storage with less overhead, but it also brings many potential threats, for example, data integrality, data availability and so on. In this paper, we propose a new remote integrality and availability checking scheme for cloud storage. This new scheme can check mass file's integrality and availability with less storage, computation and communication resource. The new scheme also supports data dynamic update, public verifiability and privacy preserving.

Key Words:cloud storage; data security; data integrality

1 Introduction

Advances in networking technology and increase in the need for computing resources have prompted many organizations to outsource their storage needs[1].This new storage model is commonly referred to as cloud storage, which has the same characteristics as cloud computing in terms of agility, scalability, elasticity. Now, there are many well know cloud storage service, for example, Microsoft's Azure storage service, Amazon's S3, Google Cloud Storage and so on.

Cloud storage service has many advantages[2]. With this service, users do not need to install physical storage devices in their own data-center or offices. They can get a myriad of virtual storage resources that might not be affordable just from a common Web browser, and don't need a complete understanding of the infrastructure. In addition, the storage maintenance tasks, such as backup, data replication, and purchasing additional storage devices are offloaded to the responsibility of a service provider, allowing organizations to focus on their core business. It seems that the only thing users have to do is to pay for this service ,which is always very cheap.

While the benefits of using a cloud storage are clear, it introduces significant security and privacy risks. In cloud storage, the data is uploaded to the cloud without leaving a copy in their local computers. This feature means that the control of the data has exceeded user's own physical worlds. To advance the adoption of cloud storage, proactive measures must be taken to ensure data security in this new environment. In fact, it seems that one of the biggest hurdle is concern over the integrity of data[1][2].

In order to solve remote integrity checking problem in cloud storage, a lot of works have been done[3,4,5,6,7,8,9,10,11,12,13,14], focusing on various conditions of application and attempting to achieve different goals. Ateniese et al.[4][8] define the "provable data possession"(PDP) model for ensuring possession of files on un-trusted storages. Their scheme utilizes the RSA-based homomorphic authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. However, the schemes of Ateniese et al. have to set a prior bound on the number of audits and doesn't support public audit ability. Juels et al.[5] describe a "proof of retrievability"(POR) model and give a more rigorous proof of their scheme.Their scheme use

* This work is supported by NSF 8632009aa012201 and the project of Modern Communications Laboratory(No. 9140C1103040902).

spot-checking and error-correcting codes to ensure both "possession" and "retrievability" of remote data files. But these operations make the number of challenges is a fixed prior, prevent any efficient extension to support updates and public audit ability. Erway et al.[10] was the first to propose dynamic PDP scheme.They developed a skip lists based method to enable provable data possession with dynamic support. however, the efficiency of their scheme remains in question. In [11],Wang et al. provided a dynamic architecture for public checking. The challenge-response protocol in their paper can both determine the data correctness and locate possible errors. However, the inefficient performance greatly affects the practical application of their scheme.

while all above schemes provide methods for assurance on the correctness of remotely stored data, none of them integrate efficient checking, dynamic update, public verifiability and privacy preserving for data storage in cloud.That is the problem we are going to tackle in this paper.

The rest of the paper is organized as follows. Section 2 introduce the architecture of the scheme, design goals and the adversary mode. Section 3 provide the detailed description of our scheme. The security analysis and performance evaluation is given in the section 4. Finally, section 5 conclude the remark of the whole paper.

2 Architecture, Design Goals and Adversary Model

2.1 Architecture of the scheme

Here, we use a basic architecture which is partly as the same as[1]. At its core, the architecture consist of three parties:a data owner (DO), that creates data and processes it in some way before it is sent to the cloud; a data auditor (DA), that checks whether the data in the cloud has been tampered with; a cloud storage service provider (CSSP), that provides cloud storage service to the cloud user.

To use the integrity checking function, DO should begin by downloading a application that consists of a data processor and a data verifier. Upon its first execution, DO generates a key, which is used to check integrity of its data, support public verifiability and privacy preserving. DA is a third party auditor, whom is authorized by DO to verify the integrity of cloud data on demand without retrieving a copy of the whole data. Whenever DO wants to upload data to the cloud, the data processor is invoked. It computes some integrity-checking value, which is used to check the integrity of the remote data, then sent the data to the cloud. Whenever DO(or DA) wants to verify the integrity of her data, the data verifier is invoked. It interacts with CSSP and ascertains the integrity of the data.

The architecture of the scheme is illustrated in Figure 1. In addition, it is important that all the core components should be either open-source or verified by the third trusted party for security.

2.2 Design Goals

(1) Integrity checking: the proposed scheme should correctly check the remote data integrity in cloud storage without a copy of the data in local.

(2)Security analysis: give a security analysis of the proposed scheme, which shows that it is secure against the un-trusted server and adversary described below.

(3)Privacy preserving, public verifiability and dynamic data operation support: The design should be as efficient as possible to ensure the seamless integration of privacy preserving, public verifiability and dynamic data operation.

(4)efficiency: The theoretical results should demonstrate that our protocol is efficient.

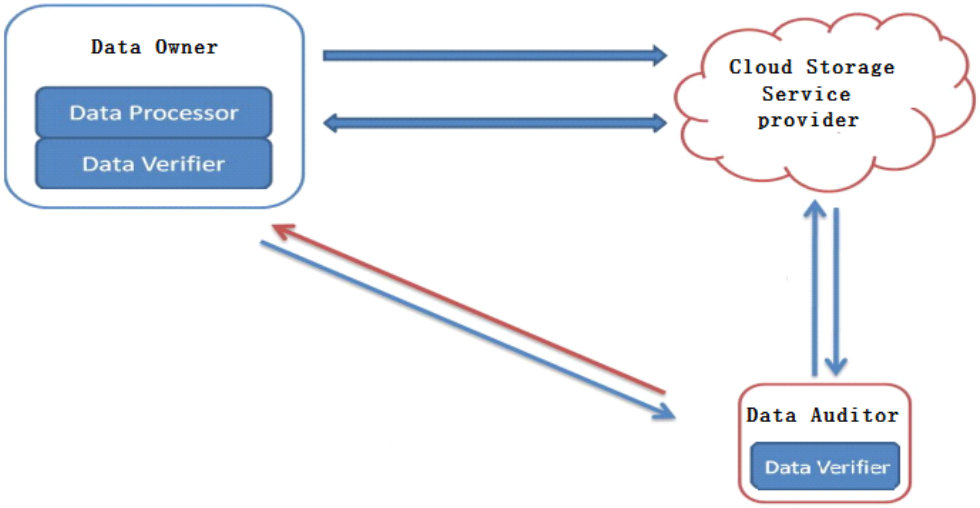

Figure 1：architecture of the scheme

2.3 Threat Model

In our scheme, we assume that the security threats come from two different sources, CSSP and malicious adversary. On the one hand, the malicious adversary may have knew all the details of the scheme, can quietly pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the DO. On the other hand, the CSSP is self-interested, un-trusted. It may hide the data corruptions caused by server hacks or Byzantine failures to maintain reputation. Moreover, wo also assume that all the communication channels between each other is security and the key stored locally on DO or DA is secret from the CSSP and adversary.

3 The Proposed Scheme

In this section, we describe our integrity checking scheme, in which the verifier stores only a single cryptographic key and a precompute value, irrespective of the size of the file it seeks to verify, as well as a small amount of some dynamic state. Firstly, we present the definitions for the necessary expressions. Secondly, a basic integrity checking scheme is proposed for single file. Then, we extent our basic scheme to support batch files checking. Finally, we describe the integrity checking in privacy preserving, public verifiability and some dynamic states.

3.1 definition of expression

$Pad(m, l)$:This function is invoked by the DO before upload. It cuts the data file m into many

blocks. The length of each block is l. If the last block is short than l,fill with 0 for padding.

$KeyGen(1^l) \to (pk, sk)$:This is a probabilistic algorithm run by the DO to generate keys. It takes as input a security parameter 1^l (l-bits), and outputs a public key pk and a secret key sk . With given parameter 1^l , it randomly chooses two distinct large prime numbers p and q ,such that $p^2 \mid q-1$. Let G be the p^2 order cyclic subgroup of Z_q . Assume that g is the generator of G . Then, it randomly Chooses two nonzero integers e and d , $e \neq d$. $pk = q$ is public to everyone ,while $sk = (e, d, g, p)$ is kept secret by the verifies.

$GenCheckValue(m, sk) \to (f_1(m), f_2(m))$:This function is used by the DO to generate a checking value, which is used for verify the response from the CSSP for future. For the data file $m = m_1 \cdots m_n$, computes:

$$f_1(m) = g^{\sum_{i=1}^{n} e^i m_i}$$

$$f_2(m) = g^{\sum_{i=1}^{n} i e^{i-1} p m_i}$$

$Challenge(sk) \to chal$:This is a probabilistic procedure run by the verifiers(DO or DA) to create a challenge for the CSSP. It takes as input secret key sk , and outputs a challenge value $chal$ .With given $sk = (e, d, g, p)$, the function picks two random numbers, r_1 and s, computes $chal$:

$$chal = (g^s, x = e + r_1 pq + dp^2)$$

$GenResponse(pk, chal) \to y(m)$:This function is used by the server CSSP to generate a response to a challenge value $chal$. Given data file $m = (m_1, \cdots, m_n)$, the CSSP computes :

$$y(m) = (g^s)^{\sum_{i=1}^{n} x^i m_i}$$

$Verify(sk, f_1(m), f_2(m), y(m)) \to \{true, false\}$:This function is used by the verifiers(DO or DA) to check the validity of the response. The verifiers compute $(f_1(m)(f_2(m))^{r_1})^s$, if $(f_1(m)(f_2(m))^{r_1})^s = y(m)$, output " $true$ ", otherwise output " $false$ ".

3.2 our integrity checking scheme

3.2.1 basic scheme

Assume that DO want to upload the file m to the cloud and check the integrity latter. The

security parameter is noted as l. Then, our basic scheme can be constructed in two phases, **Setup** and **check.**

Setup: As initialization, DO generates the public key pair (pk, sk) by executing $KeyGen(1^l) \to (pk, sk)$, sends pk to the CSSP and keeps sk secret. Once DO wants to upload file m, they invokes the function $Pad(m, l)$ to cut file m into a ordered collection of l isometric blocks, assume the result is $m = m_1 m_2 \cdots m_n$. Then, they computes the checking-value $(f_1(m), f_2(m))$ of m by using $GenCheckValue(m, sk)$, which is secretly kept in local.

Check: The verifiers (DO or DA) use $Challenge(sk) \to chal$ to generate the challenge value $chal$, send it to the CSSP. Once the CSSP get the challenge value $chal$, they invoke $GenResponse(pk, chal)$ to generate response value $y(m)$. After receiving the corresponding response value $y(m)$, the verifiers check the response by invoking $Verify(sk, f_1(m), f_2(m), y(m))$. If the output is $true$, then we believe that the file m is stored correctly in the remote servers, also we believe not.

3.2.2 batch of files checking scheme

The integrity checking scheme of batch of files is extend over the basic scheme. Assume that we have to check files which is a finite ordered at one time, for example $M = \{M_1, M_2, \cdots, M_h\}$, for each $i \in \{1,2,\cdots,h\}$, the file M_i can be padded into l_i blocks, that is: $M_i = m_{i1} m_{i2} \cdots m_{il_i}, 1 \le i \le h$. Denoted that the checking value of file M_i is($f_1(M_i)$, $f_2(M_i)$), which is computed before upload, the blocks of all checking files is $H = \sum_{i=1}^{h} l_i$. Then the check process is operate as blow:

The challenge value is generated as the same as the basic scheme. When the CSSP receive the challenge value $chal$ form verifiers, they compute the response like that:

$$Y(M) = (g^s)^{\sum_{i=1}^{h} x^{\sum_{k=1}^{i-1} l_k} \sum_{j=1}^{l_i} x^j m_{ij}}.$$

Once the verifiers receive the the response $Y(M)$, they consider that whether $Y(M) = (\prod_{i=1}^{h} f_1(M_i)^{e^{\sum_{j=1}^{i-1} l_j}} (\prod_{i=1}^{h} f_2(M_i)^{e^{\sum_{j=1}^{i-1} l_j} + \sum_{j=1}^{i-1} l_j})^{r_1})^s$ is equal to $Y(M)$. If not, they believe that the files they have checked is not correct now.

3.3 dynamic update

In practical scenarios, the client may frequently perform block-level operations on the data files.The most general forms of these operations we consider in this paper are modification, appending.

Block modification: Assume that DO wants to modify the *i*-th block m_i of her file *m*. Denote the modified data block by m_i^* and the corresponding modified checking value by $f_1^{update}(m)$, $f_2^{update}(m)$. Then, DO modify the checking value like this:

$$f_1^{update}(m) = f_1(m)g^{e^i m_i^*} / g^{e^i m_i}$$

$$f_2^{update}(m) = f_2(m)g^{ie^{i-1} pm_i^*} / g^{ie^{i-1} pm_i}$$

Block appending: Assume that DO wants to append the block m^* to her file $m = m_1 \cdots m_n$. Then, DO modify the checking value like this:

$$f_1^{update}(m) = f_1(m)g^{e^{n+1} m^*}$$

$$f_2^{update}(m) = f_2(m)g^{(n+1)e^n pm^*}$$

3.4 Public Verification and Privacy Preserving

For public verification, the DO generate the keys (pk, sk) and the corresponding checking value of files for DA, which is processed as the same as itself. Then DA use these value to make the checking scheme as above. In addition, because the challenge value $chal$ is chosen randomly and the response value $y(m)$ or $Y(m)$ will not leak the information of the files(for the difficulty of the discrete logarithm). So, we believe that our scheme can support public verification and privacy preserving.

4 Security and Efficiency Analysis

In this section, we present a analysis of the security and the efficiency of our proposed scheme. Our security analysis focuses on the adversary model define in section 2. The evaluation of efficiency is presented via comparison with other scheme.

4.1 security analysis

Generally, the checking scheme is secure if there exists no polynomial-time algorithms that can

cheat the verifier with non-negligible probability. All our proofs are derived on the probabilistic base with probability assurance.

Theorem 1: For file $m = m_1 \cdots m_n$, if the problem of discrete logarithm and big-integer factorization is difficult, then our schemes can guarantee the correctness of checking with probability $\frac{1}{q}$, while q is the public key.

Proof:

For basic scheme, when DO or DA check the response $y(m)$, they process like this:

$$y(m) = (g^s)^{\sum_{i=1}^{n} x^i m_i} = (g^{\sum_{i=1}^{n} x^i m_i \bmod p^2})^s = (g^{\sum_{i=1}^{n} e^i m_i} (g^{\sum_{i=1}^{n} i e^{i-1} pq m_i})^{r_1})^s = (f_1(m)(f_2(m))^{r_1})^s$$

From the expression of above, we can know that if the file is correctly stored, the response from the honest CSSP can pass the checking scheme. If there is some wrong on the storage or the response value is forked by the CSSP, for general, we denote the wrong response as a random value $y^*(m)$, then, in the checking process, if it can pass the checking scheme, that mean:

$$y^*(m) = (f_1(m)(f_2(m))^{r_1})^s$$

For r_1 , s are chosen randomly, anybody except verifies can not get the value from the challenge response $chal$ (the difficulty of discrete logarithm and big-integer factorization).We know that the probability of expression above is $\frac{1}{q}$.

For the batch checking scheme, the proof is the same as above, it is omitted for short. #

4.2 Performance Analysis

In this section, we consider the communication, storge and computation costs of our scheme. The detail description of the results is demonstrated in figure 2:

parameters / scheme	Computation	Comm	storage	Public Verification	Privacy Preserving	Dynamic operation
Our scheme	*O(log n)*	*O(1)*	*O(1)*	Yes	Yes	No insert No delete
Ateniese[2]	*O(1)*	*O(1)*	*O(1)*	Yes	Yes	Append only
Erway[5]	*O(log n)*	*O(log n)*	*O(1)*	No	Yes	All support
Wang Qian[6]	*O(log n)*	*O(log n)*	*O(1)*	Yes	No	All support

Figure 2: efficiency comparison for a file consisting of *n* blocks

5 Conclusion

In this paper, we investigated the problem of data integrity in cloud data storage. To solute this problem, we proposed an effective and flexible challenge/response scheme with public verification, privacy preserving and dynamic data support. According to the analysis of security and efficiency, we believe that our scheme is amenable to real-world application. The data owners can have a strong evidence that their data in cloud is stored correctly by using the protocols we proposed.

References

[1] S. Kamara, K. Lauter. Cryptographic Cloud Storage[A].Lecture Notes in Computer Science. Financial Cryptography and Data Security[C].Berlin:Springer,2010:136-149.

[2] L.M. Kaufman.Data Security in the World of cloud computing[J].Security & Privacy, IEEE, 2009,7:61-64.

[3] M. Einar, N. Maithili, T. Gene. Authentication and integrity in outsourced databases[J]. ACM Transactions On Storage. 2006,2(2):107-138.

[4] G. Ateniese, R. Burns, R. Curtmola, et al. Provable data possession at untrusted stores[A]. Proceedings of CCS 2007[C]. Alexan-dria, VA, USA, 2007.598-609.

[5] A. Juels, B. Kaliski. Pors: proofs of retrievability for large files[A]. Proceedings of CCS 2007 [C]. Alexandria, VA, USA, 2007. 584-597.

[6] H. Shacham, B. Waters. Compact proofs of retrievability[A].Proceedings of ASIACRYPT 2008[C]. Melbourne, Australia, 2008.90-107.

[7] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, et al. Efficient Remote Data Possession Checking in Critical Information Infrastructures. IEEE Trans. Knowledge and Data Eng., 2008.1034-1038.

[8] G. Ateniese, R.D. Pietro, L.V. Mancini, et al. Scalable and efficient provable data possession [A]. Proceedings of the 4th international conference on security and privacy in Communication networks[C]. Istanbul, Turkey:ACM, 2008.

[9] C. Wang, Q. Wang, K. Ren, et al.. Ensuring data storage security in cloud computing[A] Proceedings of IW QoS 2009[C], Charleston, South Carolina, USA, 2009.

[10] C. Erway, A. Kupcu, C. Papamanthou, et al. Dynamic provable data possession [EB/OL] Cryptology ePrint Archive, Report 2008/432, 2008.

[11] Q. Wang, C. Wang, J. Li, et al. Enabling public verifiability and data dynamics for storage security in cloud computing[J]. Lecture Notes in Computer Science, 2009, Volume 5789/ 2009:355-370.

[12] K.D. Bowers, A. Juels, A. Oprea. Proofs of retrievability: Theory and implementation.[A] In: Proc. of the 2009 ACM Workshop on Cloud Computing Security, CCSW 2009, Co-Located with the 16th ACM Computer and Communications Security Conf., CCS 2009. New York: Association for Computing Machinery, 2009. 43−54.

[13] S. Kamara, K. Lauter.Cryptographic Cloud Storage[A].Lecture Notes in Computer Science. Financial Cryptography and Data Security[C].Berlin:Springer,2010:136-149.

[14] Z. Hao, N. Yu. A multiple-replica remote data possession checking protocol with public verifiability[A]. Proc. Second Int'1 Data, Privacy and E-Commerce Symp. (ISDPE '10),2010.