International Journal of Future Generation Communication and Networking
Vol. 8, No. 6 (2015), pp. 115-122
http://dx.doi.org/10.14257/ijfgcn.2015.8.6.11

A Remote File Possession Checking Protocol in a Cloud Storage

Zuojie Deng[1, 2] and Xiaolan Tan[1]

[1] *School of Computer and Communication, Hunan Institute of Engineering,Xiangtan, Hunan, 411104, China*
[2] *School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei, 430074, China*
zuojiedeng@gmail.com, xltdby@126.com

Abstract

Outsourcing a file to a remote cloud storage provides several benefits, including scalability, accessibility, data replication and considerable cost saving. Unfortunately, when we send a file to a remote cloud storage server, we do not know if this file is intact. To address this problem, we present a remote file possession checking protocol. In this protocol, we use three technologies: Firstly, we design a kind of file block tag whose security is based on the discrete logarithm problem (DLP); secondly, we use probability checking method to improve its effectiveness; thirdly, we use tag aggregation method to reduce its communication cost. Our theory analysis and experiment results show that our protocol is practical and secure.

Keywords: *File possession checking, Provable security, DLP, Cloud storage*

Nomenclature

pk	public key
sk	private key
σ_i	tag of file block i
L	length of a file block
b_i	file block i
$(b_1,...,b_m)$	vector of file block
p,q	large primes
r_i,u_1,u_2,h_1,h_2	random numbers
Z_P*	multiplicative group of integers modulo p
m	number of challenge block
1^k	security parameter
$\prod_1, \prod_2$	pseudorandom permutations
K_1, k_2	keys of pseudorandom permutations $\prod_1, \prod_2$
$(c_1,\ldots c_m)$	vector of pseudorandom coefficient

1. Introduction

With the wide application of information technology, we have stored and maintained a lot of valuable files, such as documents, emails, photos, videos, and so on. To share and maintain these files easily, we usually outsource them to some remote cloud storages. Cloud storage denotes a family of increasingly popular on-line services for archiving, backup, and even primary storage of files. Amazon's S3 [1] is a well-known example. Using cloud storage to share and store files provides several benefits, such as scalability, accessibility, data replication and considerable cost saving [2]. Unfortunately, after we have sent a file to a remote cloud storage server, we do not know whether this file is intact. In 2008, Amazon's S3 suffers downtime for several hours, which cause many users unable to access their files remotely [3]. Moreover, some cloud storage providers only consider their economic benefit, they will delete some files that hardly be accessed by their users, to save their storage space. To check the status of our outsourced files, we can download all these files from the cloud storage to local disk and check them. However, this method is not practical, it wastes tremendous network bandwidth. Conventional integrity checks, for instance, cyclic redundancy check (CRC), is useful to detect accidental integrity loss. But, these integrity checks cannot detect malicious integrity attacks. If the cloud storage provider can control the communication channel, it can replay the CRC of a previous version of a file, different from the current one.

Remote possession checking protocols have been proposed in the last few years [4-5]. Using one of such protocols, a cloud provider can convince a user that the cloud provider has stored his files in a complete and uncorrupted status. However, the efficiency of all these protocols is low. To address this problem, we present an efficient remote file possession checking protocol. In this protocol, we use three technologies: Firstly, we design a kind of file block tag whose security is based on the discrete logarithm problem (DLP); secondly, we use probability checking method to improve its effectiveness; thirdly, we use tag aggregation method to reduce its communication bandwidth cost. We summarize our contribution as follows:

Firstly, we propose an efficient remote file possession checking protocol based on DLP, and improve its effectiveness by virtue of using probability checking method and tag aggregation technology. In the following section, we call it EFPCP for short. Secondly, we prove that EFPCP is secure from the theory field. Thirdly, we do some experiments for the EFPCP. The experiment results show that our EFPCP is secure and practicable.

We organize the rest of the paper as follows. In Section 2, we describe the system model and the problem. In Section 3, we present an efficient remote file possession checking protocol. In Section 4, we give some rigorous proofs for the security of this protocol. In Section 5, we do some experiments for the EFPCP, and the experiment results are presented here. Finally the conclusions and the future work are given in Section 6.

2. The System Model and the Problem

The system model is described in Figure 1. There exist three kinds of entities in the cloud storage system, a cloud storage provider, some users, and a cloud storage which is made up of some storage servers. The cloud storage provider owns and manages these cloud storage servers. A user outsources his files to a cloud storage server first and entrusts the cloud storage provider to maintain his files, then deletes his files from his local disk. In the subsequent time, the user checks the possession status of his files remotely.

In our cloud storage system model, the cloud storage provider only considers his economy benefit, he will delete some file blocks that hardly be accessed by his users to save his storage space. In the subsequent time, he tries to use some tricks to cheat his user that he preserves these files intact. In the following section, we regard the malicious cloud storage provider as an adversary, and his goal is to pass the file integrity checking without access the file completely. Note that in the following section, we use the cloud storage provider and cloud storage server interchangeably. The file possession checking problem can be described formally as follows:

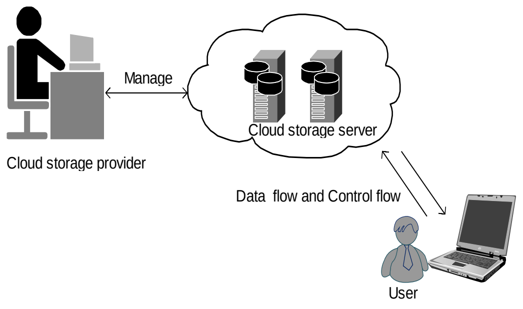


Figure 1. The Cloud Storage System Model

When a user decides to outsource a file F to the cloud storage, he generates a pair of public key and private key (pk, sk), and keeps the sk secret, and publish pk publicly. He uses the private and public key pair to generate some tags of F. We denote the tags as σ_i ($1 \le i \le n$), where n=len(F)/L, L is the length of the file block. Finally, the user sends F and σ_i ($1 \le i \le n$) to the cloud storage. After that, the user checks the possession of F. That is to say, the cloud storage server cannot cheat the user successfully without being found. In this work, we regard all these files as a collection of blocks. That is, $F=b_1b_2\dots b_n$. The length of the blocks is L bytes, which can be 128 bytes, 256 bytes or 512bytes etc. If the length of the last block doesn't equal L, it will be padded with some 0s.

3. A Remote File Possession Checking Protocol

In the previous section, we have defined the remote file possession checking problem in a cloud storage system. In this section, we design the remote file possession checking protocol whose security is based on the discrete logarithm problem (DLP). The protocol consists of the following algorithms:

(1) Keygen (1^l) → {sk,pk}. This algorithm is run by the user. It takes the security parameter 1^l as input and outputs the secure key sk and the public key pk. Its concrete algorithm steps can be described as follows:

① Let q be a l-bit prime, and p be another large prime such that q|(p-1).

② Select h_1 and h_2 uniformly at random from z_p^* such that the order of h_1 and h_2 are q.

③ Select u_1, u_2 uniformly at random from z_p^* and set $h = h_1^{u_1} * h_2^{u_2} \bmod p$.

④ Let sk={(p,k,u_1,u_2)} and pk={(p,h_1,h_2,h,g^k)}

⑤ The coefficient domain C is [0, q] and the block space is $B=[0,q]^m$

(2) GenTag(sk,b_i) →{σ_i}. This algorithm is run by the user. It takes sk and b_i as input and outputs the verification tag of file block b_i($1 \le i \le n$). For $b_i \in B$, the tag σ_i is computed by selecting r_i ($1 \le i \le n$) uniformly at random from z_q^*. $\sigma_i=(x_i,y_i)$, where $x_i = (h)^{r_i} \bmod p$, and $y_i = r_i - b_i \cdot g^k \bmod q$.

(3) Genchal (1^k)→{k_1,k_2,m}. This algorithm is run by verifier. In following section, the verifier denotes the user, the prover denotes the cloud storage server. To improve its efficiency, we use the probability checking method to select some challenged blocks. That is to say, to check the possession of file F, the verifier challenges the cloud server to prove possession of a random subset of blocks of F. It takes the security parameter 1^k as input and output k_1, k_2, and m. The verifier's challenge to the cloud server is made up of k_1, k_2 and m. m is the number of challenged blocks. k_1 is the key of the pseudorandom permutation $\prod_1$ which determines the indexes of the challenged blocks and k_2 is the key of the

pseudorandom permutation $\prod_2$ which is used to determine some random numbers. Here k_1 and k_2 are chosen for each challenge randomly.

(4) GenProof(k_1,k_2,F,m,σ_i) $\rightarrow$\{(x,y,b)\}. This algorithm is run by the prover. To reduce the network bandwidth, we use the tag aggregation method to aggregate the challenged blocks and tags in the proof. Once the sever receives a challenge, it uses the pseudorandom permutation $\prod_1$ keyed with k_1 to determine the index i of the challenged blocks, where i=$\prod_{1,k1}$(j) ($1\leq j\leq m$), and uses the pseudorandom permutation $\prod_2$ keyed with k_2 to generate these pseudorandom coefficients c_i, where c_i=$\prod_{2,k2}$(j) ($1\leq j\leq m$). Then the server generates a proof of possession R=(x,y,b) for these blocks and return it to the challenger, and x,y,b satisfies the following formulas respectively.

$$x = \prod_{i=1}^{m} (x_i)^{c_i} = \prod_{i=1}^{m} (h^{r_i})^{c_i} = h^{\sum_{i=1}^{m} c_i r_i} \bmod p \tag{1}$$

$$y = \sum_{i=1}^{m} c_i y_i = \sum_{i=1}^{m} (r_i - b_i g^k) c_i = \sum_{i=1}^{m} r_i c_i - g^k \sum_{i=1}^{m} c_i b_i \bmod q \tag{2}$$

$$b = \sum_{i=1}^{m} c_i b_i \bmod q \tag{3}$$

(5) VerifyProof((k_1,k_2,x,y,b))$\rightarrow$\{('true','false')\}. Upon receiving the responses from the sever, the challenger checks if x equals $h^{(y+g^k.b)}$. If $h^{(y+g^k.b)}$ equals x then it outputs 'true', otherwise it outputs 'false'. It can be verified that $b = \sum_{i=1}^{m} c_i b_i$ matches the aggregated tag σ because:

$$\begin{aligned} h^{(y+g^k.b)} &= h^{(\sum_{i=1}^{m} r_i c_i - g^k \sum_{i=1}^{m} c_i b_i + g^k \sum_{i=1}^{m} c_i b_i)} \\ &= h^{\sum_{i=1}^{m} r_i c_i} \\ &= x \end{aligned}$$

4. Security Analysis

From previous discussion, we know that the cloud storage provider is always selfish. He only considers his economy benefit; he will delete some file blocks that are hardly accessed by the user to save his storage space, and tries to use some tricks to cheat the user that all these files are intact. Hence, the goal of our protocol is try to catch all these cheating acts which are captured by Theorem 1.

Theorem 1. Under discrete logarithm problem (DLP) is hard, no cloud storage providers can pass our protocol in polynomial time without access the file completely.

Proof. The malicious cloud storage provider acts as the prover *A* and the user acts as the challenger. We show that no *A* can win the following security game with a non-negligible probability.

The security game is described as follows:

Setp1. The challenger runs KeyGen(1^l) to generate a public-private key pair (pk,sk), and gives pk to *A*.

Setp2. *A* may adaptively make oracle queries the tag σ_i of b_i. The challenger computes σ_i ($1 \le i \le n$) and returns σ_i ($1 \le i \le n$) to *A*. *A* keeps a list of blocks and tags: $(b_1,\ldots,b_n)$ and $(\sigma_1\ldots, \sigma_n)$.

Setp3. *A* may make oracle queries to the challenger by selecting a vector of coefficients $(c_1,c_2\ldots c_m)$ and obtains the aggregated tag σ and the aggregated block $\sum_{i=1}^{m} c_i b_i$, this can be performed in polynomial many times.

Step4. Finally, *A* selects a vector of coefficients $\hat{c} = (c_1,c_2\ldots c_m)$.

Step5. The adversary *A* wins the game, if verfyproof(pk, b', σ)='true', where $b' \ne \sum_{i=1}^{n} c_i b_i$ and $\sigma= (\sigma_1,\ldots,\sigma_m)$ is the tag vector that corresponds to the block vector $(b_1,\ldots,b_m)$ and the coefficient vector $\hat{c} = (c_1,\ldots,c_m)$ is provided by the adversary *A*.

We say the protocol is secure if *A* can not win the game with a non-negligible probability in the security parameter l.

As $x = \prod_{i=1}^{m} (x_i)^{c_i} = \prod_{i=1}^{m} (h^{r_i})^{c_i} = h^{\sum_{i=1}^{m} c_i r_i} \bmod p$ and the discrete logarithm problem (DLP) is hard, we know that no adversary can forge x' that makes x' equate x in polynomial time. So we suppose the adversary *A* can construct $b' = c_1 b_1 + \ldots + c_{i-1} b_{i-1} + c_i b_i' + c_{i+1} b_{i+1} + \ldots + c_m b_m \ne \sum_{i=1}^{m} c_i b_i = b$ that passes the verification with non-negligible probability in polynomial time. Therefore according to our remote file possession checking protocol. We get $x = h^{\sum_{i=1}^{m} c_i r_i} = h^{(y+g^k.b')} = h^{(y+g^k.b)}$. That is to say, $h^{(y+g^k.b')} = h^{(y+g^k.b)}$, then we can get $h^{g^k c_i (b_i' - b_i)} = 1$.

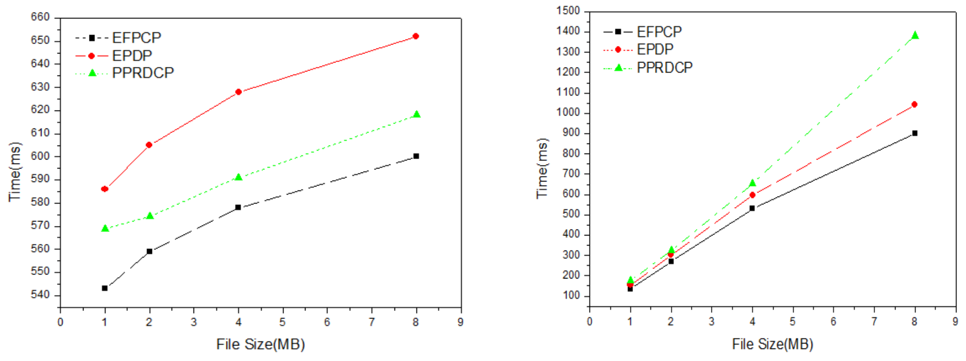
Since $(h_1^{u_1}.h_2^{u_2})^{c_i.g^k(b_i'-b_i)} = 1$, and $h_1^{u_1}.h_2^{u_2} \ne 1$, we can get $(b_i' - b_i) g^k = 0$. This is to say, $b' = b$.

From the previous assumption, we know $b' \ne b$, this is contradiction. Therefore, the adversary *A* cannot construct $b' \ne b$ that pass the protocol in polynomial time without access the file completely. This completes the proof.

5. Experiment

In this section, we evaluate the performance of our remote file possession checking protocol (EFPCP). We compare the performance of our protocol with that of the two state-of-the-art protocols [4-5]. We call the protocol in [4] E-PDP, and call the protocol in [5] PPRDCP. The experiments are running on a PC with an Intel Pentium Dual E2160 processor clocked at 1.8 GHz and a Linux Fedora operating system with kernel 2.6.23.1. The RAM of the PC is 2GB. The Disk of the machine is Western Digital Caviar SE Hard Drive that has 320GB capacity, 7200rpm with 8 MB Cache. We use the OpenSSL[6] cryptographic library (version 0.98g). In all the experiments, all the experiment data represents the mean of 10 trials.

We first evaluate the space efficiency of our protocols. To offer the same level of security, we use comparable parameters that offer the same level of security for our EFPCP, E-PDP and PPRDCP. In our EFPCP, the parameter q is 140-bit and p is 512-bit, the parameters modulus N of E-PDP and PPRDCP both are 1024-bit. The tag size in our EFPCP is 792 bits, and the tag size in [4-5] is 1024 bits. As the tags in our protocol is shorter than those in [4-5], the storage-space of our solution is more efficient than those in [4-5]. We then compare the computation performance of our EFPCP with those of E-PDP and PPRDCP. Figure 2 shows computation costs of the client and server when challenge 460 blocks for different file size. The size of the file blocks in Figure 2 is 4KB. This challenge represents 99% confidence that less than 1% of the data have been damaged [4]. Our EFPCP protocol outperforms all the other protocols in the experiment, it decreases the server computation time when a number of verifiers are connected to the server that causes enormous computation overhead over the server, and it also decreases the verifiers' computation time.



(a) Server Computation Time (b) Verifier Computation Time

Figure 2. Comparison of Computation Performance

6. Conclusion

Currently, cloud storage has become an important storage pattern and users can outsource their files there. This storage pattern provides several benefits for users, including scalability and accessibility, and considerable cost saving. It also brings some security risks to users. In this paper, we have studied the file possession checking problem. We propose a remote file possession protocol that achieves our goals. We have showed that our proposed protocol is provably security through some security analysis. Currently we are still working on improving the efficiency of our protocol and extending it to support data dynamic. We will use some other new cipher technology and some useful data structures to extend our protocol.

Acknowledgments

This work was supported in part by a Grant-in-Aid for Science and technology plan project in Hunan province (No. 2014GK3157, No. 2014FJ3054).

References

[1] "Amazon simple storage service (Amazon S3)", http://aws.amazon.com/s3/.
[2] M. Armbrust, A. Fox and R. Griffith, "Above the clouds: A berkeley view of cloud computing", Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, **(2009)**.
[3] M. Paul, "Amazon's S3 storage service suffers downtime", http://www.techrepublic.com/blog/tech-news/amazons-s3- storage -service-suffers-downtime/2064.
[4] G. Ateniese, R. Burns and R. Curtmola, "Provable data possession at untrusted stores", Proceedings of the 14th ACM conference on Computer and communications security(CCS'07), Alexandria, Virginia, USA, **(2007)** , pp.598-609.
[5] Z. Hao, S. Zhong and N.H. Yu, "A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability", IEEE Trans. Knowl. Data Eng (TKDE), vol. 23, no. 9, **(2011)**, pp.1432-1437.
[6] "OpenSSL: The Open Source toolkit for SSL/TLS", www.openssl.org/.
[7] M.A. Shah, M. Baker, J. C. Mogul and R. S. Swaminathan, "Auditing to keep online storage services honest", Proceedings of the 11th USENIX workshop on hot topics in operating systems, USENIX Association, **(2007)**, pp. 1-6.
[8] O. Goldreich, "Foundations of cryptography: Basic applications vol. 2", Cambridge University Press, **(2004)**.
[9] G. Ateniese, R. D. Pietro, L. Mancini and G. Tsudik, "Scalable and efficient provable data possession", Proceedings of the 4th international conference on Security and privacy in communication networks (SecureComm '08), ACM,New York, **(2008)**, pp. 1-10.

[10] W. J. Bolosky, J. R. Douceur, D. Ely and M. Theirmer, “Feasability of a serverless distributed file system deployed on an existing set of desktop PCs”, Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (SIGMETRICS '00), ACM Press, New York, **(2000)**, pp. 34-43.
[11] P. Maniatis, M. Roussopoulous and T. Giuli, “The LOCKSS peer-to-peer digital preservation system”, ACM Transactions on computing Systems, vol. 23, no.1, **(2005)**, pp.2-5.
[12] M. Waldman and D. Mazieres, “Tangler: a censorship-resistant publishing system based on document entanglements”, Proceedings of the 8th ACM conference on Computer and communications security (CCS '01). ACM Press, New York, **(2001)**, pp. 126-135.
[13] P. S. Kumar and R. Subramanian, “Homomorpic Distributed Verification Protocol for Ensuring Data Storage Security in Cloud Computing”, Information-An International Interdisciplinary Journal, vol. 14, no.1, **(2001)**, pp. 3465-3476.
[14] M. Bellare and O. Goldreich, “On defining proofs of knowledge”, Advances in Cryptology—CRYPTO’ 92, Lecture Notes in Computer Science 740, Springer, **(1993)**, pp. 390-420.

Authors

Zuojie Deng, he is born in September 1972, received the M.S. degree in Department of Computer Engineering from Hunan University, Changsha, China, in 2004. He is an associate professor in School of Computer and Communication, Hunan Institute of Engineering and he is a Ph.D. candidate in the School of Computer Science and Technology at Huazhong University of Science and Technology, Wuhan, China. His research areas include storage security, searchable encryption, network security and privacy enhanced technologies.

Xiaolan Tan, she is born in July 1973, and received the bachelor in computer science from Hunan Normal University in 2004 and he is an instructor in School of Computer and Communication at Hunan Institute of Engineering, Xiangtan, China. Her research interests include network security and information privacy.