A TPA Based Efficient Non-Repudiation Scheme for Cloud Storage

Wenqi Ma[1, a], Qingbo Wu[2, b], Yusong Tan[3, c]

Computer Collage, The National University of Defense Technology, Changsha City, Hunan Prov.
[a]wqma1984@gmail.com, [b]wu.qingbo2008@gmail.com, [c]yusong.tan@gmail.com

Keywords: Non-Repudiation, Cloud Storage, Cloud security, Third Party Auditor.

Abstract. One of differences between cloud storage and previous storage is that there is a financial contract between user and the cloud service provider (CSP). User pay for service in exchange for certain guarantees and the cloud is a liable entity. But some mechanisms need to ensure the liability of CSP. Some work use non-repudiation to realize it. Compared with these non-repudiation schemes, we use third party auditor not client to manage proofs and some metadata, which are security critical data in cloud security. It can provide a more security environment for these data. Against the big overhead in update process of current non-repudiation scheme, we propose three schemes to improve it.

Introduction

With the development of cloud computing, the security issues in cloud computing get more and more attention. Solving these issues is important to popularize of cloud computing. Many works solve these issues based on client. Although these works can provide security guarantee to data, it also result in a heavy burden to client. Financial contract between user and CSP is one of characters of cloud storage. User pay for service in exchange for certain guarantee and cloud should protect data in its best. But there need some mechanisms to ensure CSP practicing its liability. Several papers use non-repudiation protocol to realize it, but they use client to conserve proofs. It will result in some problems. In this paper, we use third party auditor (TPA) to manage proofs and some metadata, which are security critical data in cloud security. It can provide a more security environment for these data. For reducing the overhead of update, we propose three schemes.

Related works

Architectures of Secure Cloud Storage. For protecting data in cloud, several architectures have been proposed, such as single cloud [3] and multi-cloud [4,5]. The problem of the two architectures is it will result in a heavy burden to client. Twin-cloud [6] can solve this problem. It includes a trusted cloud and a commodity cloud, and trusted cloud assumes most of security works.

Proof of Storage and Reed-Solomon Code. Proof of storage [8,9] is a technology which allows user or third party to verify integrity of data which store in cloud. The algorithms of it can be classified into private verification and public verification [10]. Public verification can allow authorized third party to verify integrity. It can reduce the workload of client. Reed-Solomon code is a commonly used technology in storage and communication. Through dispersing encoded data to several servers, it can resist some of servers corrupt at the same time in cloud storage.

Non-Repudiate Scheme in Cloud Storage. Ada Popa et al [2] and Feng et al [1] present non-repudiate scheme in cloud storage separately. But they conserve proofs using cloud and client. It will result in some problems. For example, user accessing cloud using different client will result in proofs distributing in different client. Client also can’t provider a security enough environment for proofs. The overhead of updating process in their schemes is high. Updating a file need CSP to return a new proof. It needs CSP to sign the digest of new file. User also need download the whole file to verify the correctness of proof.

A TPA Based Non-Repudiation Scheme

Basic Model of TBNR. One character of our scheme is using TPA to manage proofs and some metadata. TPA in this paper is similar to TTP. It is an authoritative entity and acknowledged by user and CSP. The different between them is that TTP only provide service exchanging signatures and TPA need to record and manage these signatures in this paper.

TPA managing proofs and some metadata has the following advantages: First, user accessing cloud using different clients will not result in proofs dispersal in different clients. Second, it can provide a more security environment for proofs. Third, the deleting process only needs TPA to delete corresponding proof. CSP don't need to conserve all proofs deleting file forever.

In this paper, we only consider entities user, CSP and TPA. The relationship among them can be represented by the following Fig. 1:

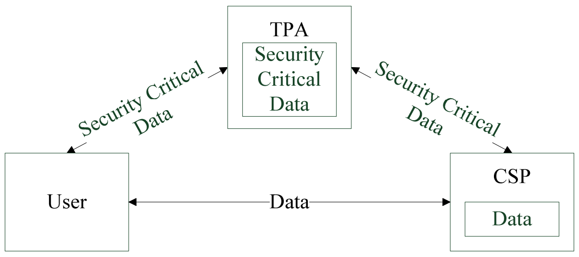

Fig. 1: Relationship among user, CSP and TPA

User and CSP exchange data directly, and exchange security critical data such like proofs through TPA. TPA records these security critical data. In this model, User, CSP and TPA are restricted each other. None of them have an absolute authority. Transactions between user and CSP are audited by TPA, malicious behavior of user and CSP will be discovered by TPA. User and CSP also can conserve some proofs to discover malicious behavior of TPA.

Basic Process of TBNR. The basic operations of cloud storage include uploading, downloading, updating, deleting and getting directory. For preventing CSP to cheat user using a wrong directory, TPA maintains the directory and user downloads it from TPA. The processes of these operations are described as follows:

Directory getting: User get directory from TPA. TPA sends the directory to user in the form of ciphertext. For reducing overhead, user can cache the directory in the client and use some mechanisms to keep its consistency.

Uploading: The uploading session can be divided into four steps. First, user sends file F and signature S_U to CSP. S_U is the signature of F's digest. CSP verify the integrity of F and resign the digest getting S_{CSP}. Second, CSP sends S_{CSP} to TPA. Third, TPA sends S_{CSP} to user to verify the correctness of S_{CSP}. Last, user tells TPA the result of verification. If verification succeeding, TPA conserves the proof and the uploading session finished.

Downloading: The downloading session is relatively simple. User downloads file from CSP and downloads corresponding signature from TPA. Here the signature is using to verify the integrity of file.

Deleting: User sends the path of file and its signature to CSP. CSP verifies the path. If verification succeeding, CSP deletes the corresponding file and transfers the message to TPA. TPA does the same verification and deletes the corresponding proof.

Updating: Before updating file, user needs to download it. Then user sends the increment ΔF to CSP. CSP signs $F+\Delta F$ and sends the signature to TPA. TPA transmits it to user to verify. If the signature is correctness, TPA conserves the proof and updating session is finished.

Data Structure of TBNR. For a description of our non-repudiation protocol, some notation and definitions are used as follows:

F: File user want to store in cloud.

P: Pathname of file in cloud. Here the P is stored and transmitted in the form of ciphertext.

T: Timestamp. It is the unique identification of one session which can prevent replay attack.

Flag: It is used to specify the type operation type. Here use "upl" to express uploading, "dow" to express downloading, "upd" to express updating, "del" to express deleting and "gdir" to express getting directory.

H(M): Digest of message M.

$S_X(M)$: Signature of message M. The key is X's private key.

$E_X(M)$: Ciphertext of message M. The key is X's key, and E is symmetric encryption algorithm.

$EA_X(M)$: Ciphertext of message M. The key is X's public key, and E is asymmetric encryption algorithm.

I. Directory getting session

Step1: U=>TPA - {T, U, CSP, gdir, $S_U(H(T|gdir))$, $EA_{TPA}(key)$}

Step2: TPA=>U - {T, U, CSP, gdir, $S_{TPA}(H(T|E_U (dir)))$, $E_{key}(dir)$}

$EA_{TPA}(key)$ is used to specify the key TPA encrypting directory, where "key" is a temporary key.

II. Uploading session

Step1: U=>CSP - {T, U, CSP, upl, P, $S_U(H(T|upl|P))$, $S_U(H(T|E_U(F)))$, $E_U(F)$}

Step2: CSP=>TPA - {T, U, CSP, upl, P, $S_{CSP}(H(T|upl|P))$, $S_{CSP}(H(T|E_U(F)))$}

Step3: TPA=>U - {T, U, CSP, upl, P, $S_{TPA}(H(T|upl|P))$, $S_{TPA}(H(T|EU(F)))$}

Step4: U=>TPA - {T, U, CSP, upl, P, SU(H(T|upl))}

III. Downloading session

Step1: U=>CSP - {T, U, CSP, dow, P, $S_U(H(T|dow|P))$}

Step2: CSP=>U - {T, U, CSP, dow, P, $E_U(F)$}

Step3: U=>TPA - {T, U, CSP, dow, P, $S_U(H(T|dow|P))$}

Step4: TPA=>U - {T, U, CSP, dow, P, $S_{TPA}(H(T|T_1|S_1))$, {T_1, S_1}}

Where $S_1 = S_{CSP}(H(T_1|E_U(F)))$

IV. Deleting session

Step1: U=>CSP - {T, U, CSP, del, P, $S_U(H(T|del|P))$}

Step2: CSP=>TPA - {T, U, CSP, del, P, $S_U(H(T|del|P))$}

Step3: TPA=>CSP - {T, U, CSP, del, P, $S_{TPA}(H(T|del|P))$}

Step4: CSP=>U - {T, U, CSP, del, P, $S_{TPA}(H(T|del|P))$}

V. Updating session

Step1: U=>CSP - {T, U, CSP, upd, P, $S_U(H(T|upd|P))$, $S_U(H(T|E_U(\Delta F)))$, $E_U(\Delta F)$}

Step2: CSP=>TPA - {T, U, CSP, upd, P, $S_{CSP}(H(T|upd|P))$, $S_{CSP}(H(T|E_U(F+\Delta F)))$}

Step3: TPA=>U - {T, U, CSP, upd, P, $S_{TPA}(H(T|upd|P))$, $S_{TPA}(H(T|E_U(F+\Delta F)))$}

Step4: U=>TPA - {T, U, CSP, upd, P, $S_U(H(T|upd))$}

From the above session we can see that the additional overhead of uploading, downloading and deleting session comparing with remote storage include digest and signature. The additional overhead of updating session not only includes digest and signature, but also includes downloading and resigning the whole file. If user doesn't download the file, he can't to verify the correctness of the new proof.

Improved Schemes on Update

In this section, we propose three improved schemes to reduce the overhead of updating session. There are block scheme, increment scheme and group scheme. We will compare their overhead in section 5.

For a description of improved schemes, we need to do some assumption on updating operation. Here we suppose the basic operation of updating is block operation which includes replacing, inserting, deleting and appending. Replacing can be seen as inserting after deleting, and appending can be seen as inserting at the end of file. So the basic block operations of updating only include inserting and deleting.

An Improved Scheme Based on Block. The non-linearity property of hash function makes CSP must resign the whole new file in updating session. User also needs to download the whole file to verify the correctness of the new signature. How to make the updating session don't involving the blocks unchanged, a directly ideal is building proof on block, not on the whole file.

Directory getting and deleting session in block scheme are same as basic scheme. The difference in downloading session is user need to verify every block separately. And the difference in uploading session is CSP need to sign every block and user needing to verify every signature separately.

The updating session in block scheme can be divided into two stages: deleting some blocks and inserting some blocks. In the deleting stage, user sends the collection of block id their signature to CSP. It just likes the process deleting file. In the inserting stage, user sends the collection of block id, block and their signature to CSP. It just likes the process uploading file. We also can merge the two stages into one.

Although the block scheme reduces the overhead of updating session, it also brings some other overheads. TPA needs to conserve more proofs because one file will corresponds to a lot of proofs. CSP needs to generate more signatures, and user needs to verify more signatures.

Another problem of block scheme is how to choose the size of block. If the size of block is too small, the number of proofs will be too many. If the size of block is too big, the overhead of updating session will increase.

An Improving Scheme Based on Increment. Block scheme results in a lot of signatures and verifications in one session. For solving this problem, a directly ideal is building proof on increment. Suppose F_0, F_1, F_2, …, F_n is a sequence of updating. F_0 is the original version of file, and $\Delta F_i=F_i-F_{i-1}$ is the increment of i-th updating. In increment scheme, user only needs to upload ΔF and verify the signature of ΔF. CSP only needs to sign the digest of ΔF. And TPA only needs to add a proof. The ΔF can be expressed as $\{del, Bid_i\} \cup \{ins, Bid_j, B_j\}$, i□D and j□I. Where D is the collection of blocks user want to delete and I is the collection of blocks user want to insert.

The advantage of this scheme is obviously. In the uploading session, CSP only needs to generate one signature. With the increase of the number of updating, the number of proofs and the total size of increments will increase. We can control the number of proofs and the total size of increments in a reasonable range through merging them into one. The merging process is download the original file and all increments to compute the latest file, and re-upload it to cloud. Obviously the overhead of merging is big.

The directory getting, uploading and deleting session in increment scheme are same as the basic scheme. The download session needs user download the original file and corresponding increments to compute the latest file. User also needs to download the corresponding proofs to verify them separately. The update session is the same as the upload session in basic scheme.

The increment scheme reduces the overhead of signature and verification comparing with block scheme when ensures the updating which only involves changed blocks. There also have some problems in increment scheme. The first is user needs to merge the original file and all increments into the latest file in downloading session. The second is when the number of increment or the total size of increments exceeding a certain threshold, user needs to re-upload the latest file.

An Improving Scheme Based on Group. In increment scheme, some blocks update frequently will result in whole file merging frequently. If the proof is building on the collection of these blocks not on the whole file, this problem seems to be solved.

The basic ideal of group scheme is divide file into some groups. Every group includes some blocks. Proof in group scheme is building on group. We use the increment method to update each group separately. File in group scheme consist of many groups and group increments. Each group and group increment has a corresponding proof.

Uploading session in group scheme is similar to block scheme. Blocks in block scheme are replaced by groups in group scheme. Deleting session is same as basic scheme. Downloading session needs to download all groups, group increments, and corresponding proofs, verify groups and group increments using proofs, and compute the latest file. Updating session can be divided into three situations which include group insert, group delete and group update. Group insert happens when a lot of continuous blocks are inserted. It just likes inserting block in block scheme. Group delete happens when a lot of continuous blocks are deleted. It just likes deleting block in block scheme. Group update happens when some scattered blocks are inserted and deleted. It is similar to updating session in increment scheme.

Comparing with block scheme, group scheme has fewer signatures and verifications. Comparing with increment scheme, group scheme avoids the situation when some blocks update frequently resulting in the whole file being merged frequently. It is the compromise of block scheme and increment scheme.

Performance and Security Analysis

Performance Analysis. Because of the locality of update, we only consider the situation when blocks changed is continuous. Suppose that the signature algorithm is ECC and the hash algorithm is SHA1. Through simple test, we find that the overhead of ECC is about 3ms and the rate of SHA1 is about 40M/s. So we assume the overhead of ECC is 3ms and the rate of SHA1 is 40M/s. Suppose that the size of file is 64M, the size of group is 1M, the size of block is 4k, the bandwidth between user and CSP is 256k/s, the number of increment doesn’t exceed 40, and the number of group increment doesn’t exceed 4. When we replace 10 blocks, 20 blocks, 30 blocks, 40 blocks, 2 groups, 4 groups, 6 groups, 8 groups, the average computation overhead of CSP and the average communication overhead between user and CSP are shown in Fig. 2.

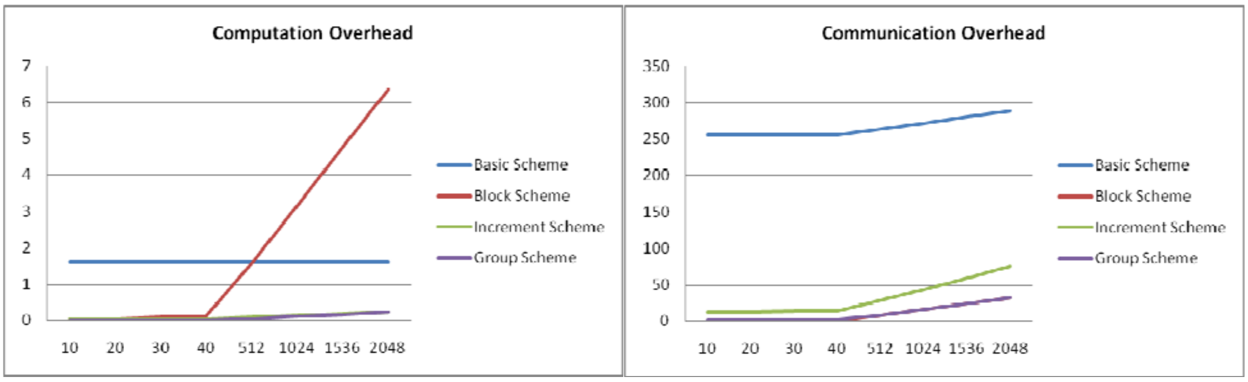

Fig 2. Computation overhead and communication overhead

From the above analysis we can see that improved schemes reduce communication overhead obviously. The communication overhead of increment scheme is greater than the two other improved schemes. And the communication overhead of the group scheme is approach to the block scheme. Improved schemes also reduce computation overhead. But when there are many blocks involved in update, the computation overhead of block scheme will be very high. So we can make a conclusion that group scheme is superior to block scheme and increment scheme when the block inserted is centralized.

Security analysis. The goal of non-repudiation is to enhance the security of cloud storage and convince potential customers that the service is secure. Therefore, it is highly desired that the TBNR protocol is robust against various threats. The reasons why our non-repudiation scheme is robust enough are as follows:

First, the confidentiality of data in TBNR protocol is protected well because file and its pathname are encrypted before uploading to the cloud. The plaintext of file only exists in client, and client only exists files user accessing.

Second, signature make the faking of message is difficult. Content of file and pathname all be signed before sending. So the attacker is difficult to implement man-in-the-middle attack.

Third, timestamp is embedded into signature which can prevent replay attack effectively. Using timestamp as the unique flag has two advantages. On the one hand, it can ensure the uniqueness of flag easily. On the other hand, we can use it to judge whether the message is time-out.

Summary and Prospect

TPA conversing proof solves the problem of proof management. User accessing cloud in different clients will not result in proofs dispersal in different clients. It also provides a more security environment for proofs. Through performance analysis, we can see that the improved scheme reduce the overhead actually. And the group scheme is superior to block scheme and increment scheme when the block inserted is centralized.

Non-repudiation scheme don't protect data integrity directly but take the task protecting data integrity to cloud. It only provides an assurance for CSP carrying out its liability. TPA in this paper is not only used in proofs management, it also can be used in other aspects. For example, third party verification in proof of storage [12].

TPA is the bottleneck of this system. So there need to make some optimizations on it. For example, user and CSP can exchange proofs directly, cache them and upload them to TPA periodically. It merges several communications with TPA into one, and can lighten the burden of TPA. We will realize it in the following works.

Reference

[1] J. Feng, Y. Chen, and Summerville, D.H. A Fair Multi-party Non-repudiation Scheme for Storage Clouds. Collaboration Technologies and Systems (CTS), 2011.
[2] R. Popa, J. Lorch, D. Molnar, H. Wang, and L. Zhuang. Enabling Security in Cloud Storage SLAs with CloudProof. Microsoft Tech Report MSR-TR-2010-46, May, 2010.
[3] S. Kamara and K. Lauter. Cryptographic Cloud Storage. Financial Cryptography and Data Security Lecture Notes in Computer Science Volume 6054, 2010, pp 136-149.
[4] A. Bessani, M. Correia, B. Quaresma, F. André, P. Sousa, L. Portugal. DEPSKY: Dependable and Secure Storage in a Cloud-of-Clouds. Proceeding EuroSys '11 Proceedings of the sixth conference on Computer systems Pages 31-46.
[5] M. AlZain, E. Pardede, B. Soh, and J. Thom. Cloud computing security: from single to multi-clouds. the 45 th Hawaii International Conference on System Sciences, Hawaii, United States, 2012.
[6] Bugiel, S., Nurnberger, S., Sadeghi, A.R., Schneide, T. Twin Clouds: An Architecture for Secure Cloud Computing. In: Workshop on Cryptography and Security in Clouds, Zurich, March 15-16 (2011).
[7] Bowers, K.D., Juels, A., Oprea, A. Hail: A high-availability and integrity layer for cloud storage. Cryptology ePrint Archive, Report 2008/489 (2008)
[8] A. Juels and Burton S. Kaliski. PORs: Proofs of Retrievability for Large Files. CCS '07 Proceedings of the 14th ACM conference on Computer and communications security Pages 584-597.
[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. CCS '07 Proceedings of the 14th ACM conference on Computer and communications security Pages 598-609.
[10] H. Shacham, B. Waters. Compact Proofs of Retrievability. Advances in Cryptology - ASIACRYPT 2008 Lecture Notes in Computer Science Volume 5350, 2008, pp 90-107.
[11] Ari Juels, Burton S. Kaliski Jr. PORs: Proofs of retrievability for large files. In: ACM conference on computer and communications security
[12] C. Wang, Q. Wang, and Kui Ren. Ensuring Data Storage Security in Cloud Computing. Quality of Service, 2009. IWQoS. 17th International.