

An Approach for Secure Data Storage Services in Cloud Computing

S. Srinivas¹, Martha Sheshikala², Duvvada Rajeswara Rao³

¹M.Tech Student, Dept of CSE, SR Engineering College, Warangal, A.P., India

²Assistant Professor, Dept of CSE, SR Engineering College, Warangal, A.P., India

³Professor, KL University, Vijayawada, A.P., India

Abstract: *A cloud storage system consists of a collection of storage servers over the internet. The main aim is to provide secure storage services in a cloud storage system. In this paper, we propose a Third Party auditing for data storage security in Cloud Computing. Encryption schemes are used to provide data confidentiality, data robustness and functionality. We use an encryption Data and Key verification for implementing for data secure storage. We guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files for cloud operations such as update, delete and so on. Extensive analysis shows that our schemes are provably secure and highly efficient.*

Keywords: Secure storage, TPA, Encryption, Data Confidentiality

1. Introduction

Cloud computing is a new computing paradigm that is built on virtualization, parallel and distributed computing, utility computing, and service-oriented architecture. In the last several years, cloud computing has emerged as one of the most influential paradigms in the IT industry, Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed. It focuses on designing cloud storage for robustness, confidentiality, and functionality. The cloud storage system is considered as a large scale distributed storage system that consists of many independent storage servers. Data robustness is a major requirement for storage systems. One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives. Another way is to encode a message of k symbols into a codeword of n symbols by erasure coding. To store a message, each of its codeword symbols is stored in a different storage server. After the message symbols are sent to storage servers, each storage server independently computes a codeword symbol for the received message symbols and stores it. This finishes the encoding and storing process. The recovery process is the same. The system model that consists of distributed storage servers and key servers. Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers that shall perform cryptographic functions on behalf of the user. The method of threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. The storage servers independently perform encoding and re-

encryption and key servers independently perform partial decryption. The parameters are flexible adjustment between the number of storage servers and robustness.

2. Problem Statement

Storing data in a third party's cloud system causes serious concern on data confidentiality. To provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. There are three problems in the above straightforward integration of encryption and encoding. First, the user has to do most computation and the communication traffic between the user and storage servers is high. Second, the user has to manage his cryptographic keys. If the user's device of storing the keys is lost or compromised, the security is broken. Finally, data storing and retrieving, it is hard for storage servers to directly support other functions. For example, storage servers cannot directly forward a user's messages to another one. The owner of messages has to retrieve, decode, decrypt and then forward them to another user. It addresses the problem of forwarding data to another user by storage servers directly under the command of the data owner.

In contrast to traditional solutions, IT services are under proper physical, logical and personnel controls, where Cloud Computing moves the application software and databases to the large data centers, where the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. The user doesn't have the privacy for preserving the data and the security risks towards the correctness of the data in cloud which may not be possible. From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing

inevitably poses new challenging security threats for number of reasons.

The traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. And the Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance.

3. Third Party Auditing

In this auditing phase, the auditor repeatedly checks the stored data using a challenge-response protocol. Each check establishes the data's integrity immediately before the check. We will also show how to handle the encrypted data and encryption key, one after the other. During auditing, the main threat which may occur in the storage service is that it lost some part of the data which can be encrypted and encryption keys added with some malicious information and can fool the auditor into believing that it has both. It may fool the auditor in two main ways (1) by manipulating the current and cached past challenges the auditor provides or (2) by combining these challenges and derived values from the data or the encryption key, such that the encrypted data and encryption key cannot be entirely recovered from the derived values. Thus, for both the encrypted data and encryption key, we need to prove two properties to ensure data integrity:

- **Completeness:** After receiving data, if the service possesses all the bits of the encrypted data and encryption key, the auditor accepts the responses.
- **Soundness:** After receiving the data, if the service is missing any bit in the encrypted data or encryption key, the auditor accepts with negligible probability.

The main threat from the auditor is that it may glean important information from the auditing process that could compromise the privacy guarantees provided by the service. For example, even a few bits from a file containing medical history could reveal whether a customer has a disease. To ensure privacy, we rely on different standards for the encrypted data and the encryption key. For the data, we rely on (1) the strength of the encryption scheme and (2) the zero-knowledge property of the protocol for encryption-key audits. Thus, we must prove the encryption-key audits with the service that can be efficiently simulated such that the auditor's interaction is indistinguishable from one with a true service.

3.1 Encrypted Data Verification

We use a simple challenge-response protocol to check the encrypted data as described in protocol.

1. A chooses any R_j, H_j from L and $L = L \setminus \{(R_j, H_j)\}$.
 1. a. $A \rightarrow S: R_j$.
2. S computes $H_s = \text{HMAC}(R_j, EK(M))$.
 - 2a. $S \rightarrow A: H_s$.
3. A checks $H_s = H_j$ else declares S lost data.

3.2 Encryption Key Verification

To determine if the encryption key is intact, we have a number of options. One option is to adapt existing identification schemes to prove the service has K without revealing K . For example, protocol uses the Schnorr identification scheme to show that the service still has K . Schnorr's scheme is complete and sound. For soundness, the service can fool the auditor into accepting with probability $< 1/2^t$. But, this protocol is only provably zero-knowledge if the auditor honestly follows the protocol.

1. A chooses a random β s.t. $1 < \beta < q$ and computes g^β .
 - 1a. $A \rightarrow S: V_a = g^\beta$.
2. S computes $W_s = (V_a)^K = g^{\beta K}$.
 - 2a. $S \rightarrow A: W_s$.
3. A computes $W_a = (g^K)^\beta$
 - 3a. A checks $W_a = W_s$ else declares S lost key.

3.3 Modules

3.3.1 System Model

- a) **User:** users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.
- b) **Cloud Service Provider (CSP):** a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.
- c) **Third Party Auditor (TPA):** an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

3.3.2 Cloud Operations

- a) **Update Operation**
In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud, we refer this operation as data update. In other words, for all the unused tokens, the user needs to exclude every occurrence of the old data block and replace it with the new one.
- b) **Delete Operation**
Sometimes, after being stored in the cloud, certain data blocks may need to be deleted. The delete operation we are considering is a general one, in which user replaces the data block with zero or some special reserved data symbol. From this point of view, the delete operation is actually a special case of the data update operation, where the original data blocks can be replaced with zeros or some predetermined special blocks.
- c) **Append Operation**
In some cases, the user may want to increase the size of his stored data by adding blocks at the end of the data file, which we refer as data append. We anticipate that the most frequent append operation in cloud data storage

is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time.

4. Results

home sign up **user** about us web service >> **USER**

[user login security](#)



Security Key [Get Security Key](#)

SUBMIT

Get Your 'Login Security Key' To Your Mail !

user home **file upload** file process my alert's sign out **Welcome kingslin !**

[User FileUpload](#)

File Name : maintain.txt Verificaton Allow/Block

First Block :

```
So far, we assumed that F represents static or archived data. This model may fit some application scenarios, such as libraries and scientific datasets.
```

--Allow-- ✓
First Block Alert Status
31167

Middle Block:

```
V05N6ecH0wqr8zmlEkjG4pe055zYWuftkTLkR5HAHDoPFjkj3Cm4PpAVocFCQiCORH8kM6s6hX8h9g3dbxhlN7QFQg1aA6FomfihJ99MJVW+E+WeaG3DmmzVdqEVHWg8fzS31e9OceNis01tMvNvCeLT7kCBeU1dUbtux9S0eO
```

--Block-- ✗
Verification Token
SDONRRBedu3+bMi1DAyuWQ==

Last Block :


```
namic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file while maintaining
```

--Allow-- ✓
Verification Token
31167

FINISH


user home file upload **file process** my alert's sign out **Welcome kingslin !**

[user file process](#)

 maintain.txt

Update Delete Append [Get First Block Token Key](#)

So far, we assumed that F represents static or archived data. This model may fit some application scenarios, such as libraries and scientific datasets. However, in and cloud data stor

Give First Block Token Key Below **SUMBIT** 

First Block File Process Success

FIRST BLOCK

Binary Data


First Block

Middle Block

Last Block

Update Delete Append [Get Middle Block Token Key](#)

age, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or files etc. Therefore, it is crucial to consider the dy

Give Middle Block Token Key Below **SUMBIT** 

Middle Block File Process Success

MIDDLE BLOCK

Update Delete Append [Get Last Block Token Key](#)

namic case, where a user may wish to perform various block-level operations of update, delete and append to modify the data file while maintaining the storage correctness assuranc

Give Last Block Token Key Below **SUMBIT**

Last Block File Process Success

LAST BLOCK

[verify file blocks](#)

conclusion.txt

FIRST BLOCK ✓

Token Key

SUBMIT **TOKEN REQUEST** **VERIFY** *First Block Verify Successfully !*

[Get First Block Token Key](#)

In this paper, we investigate the problem of data security in cloud data storage, which is essentially a distributed storage system. To achieve the assurances of cloud data integrity a



FINISH

MIDDLE BLOCK ✓

Token Key

SUBMIT **TOKEN REQUEST** **VERIFY** *Second Block Verify Successfully !*

[Get Middle Block Token Key](#)

nd availability and enforce the quality of dependable cloud storage service for users, we propose an effective and flexible distributed scheme with explicit dynamic data support, inclu

LAST BLOCK ✓

Token Key

SUBMIT **TOKEN REQUEST** **VERIFY** *Last Block Verify Successfully !*

[Get Last Block Token Key](#)

ding block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability



My Alerts !

File ID	File Name	Token Request (FB)	Modify (FB)	Token Request (MB)	Modify (SB)	Token Request (LB)	Modify (LB)	View
5	maintain.txt	NO	NO	YES	NO	NO	NO	view

FB - First Block
MB - Middle Block
LB - Last Block



[my alerts \(full view\) !](#)



File Name : maintain.txt Verificaton Allow/Block

First Block : **--Allow--** ✓

First Block Alert Status
BLOCK VERIFY

Middle Block : **--Block--** ✗

Middle Block Alert Status
BLOCK REQUEST

Last Block : **--Allow--** ✓

Last Block Alert Status
BLOCK VERIFY

FINISH

5. Conclusion

In this paper, we propose a Third Party auditing for data storage security in Cloud Computing. We use an encryption Data and Key verification for implementing for data secure storage. We guarantee that the TPA would not learn any knowledge about the data content stored on the cloud server during the efficient auditing process, which not only eliminates the burden of cloud user from the tedious and possibly expensive auditing task, but also alleviates the users' fear of their outsourced data leakage. Considering TPA may concurrently handle multiple audit sessions from different users for their outsourced data files for cloud operations such as update, delete and so on.

References

- [1] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in Proc. of IWQoS'09, July 2009, pp. 1–9.
- [2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for storage security in cloud computing," in Proc. of IEEE INFOCOM'10, San Diego, CA, USA, March 2010.
- [3] C. Wang, K. Ren, W. Lou, and J. Li, "Towards publicly auditable secure cloud data storage services," IEEE Network Magazine, vol. 24, no. 4, pp. 19–24, 2010.
- [4] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," in Proc. Of IEEE INFOCOM'09, Rio de Janeiro, Brazil, April 2009.
- [5] M. Bellare, R. Canetti, and H. Krawczyk, "Keying hash functions for message authentication," in Proc. of Crypto'96, volume 1109 of LNCS. Springer-Verlag, 1996, pp. 1–15.
- [6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. UCB-EECS-2009-28, Feb 2009.
- [7] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained access control in cloud computing,"
- [8] in Proc. of IEEE INFOCOM'10, San Diego, CA, USA, March 2010.
- [9] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS'09, 2009, pp. 213–222.
- [10] R.C.Merkle, "Protocols for public key cryptosystems," in Proc. of IEEE Symposium on Security and Privacy, Los Alamitos, CA, USA, 1980.

Author Profile



S. Srinivas Currently pursuing M.Tech in Computer Science and Engineering at SR Engineering College, Warangal, A.P., India. His research interest includes bandwidth estimation in networks, Mobile Adhoc networks and cloud computing.



M. Sheshikala has 9+ years of experience in teaching field. Her areas of interest include Data Mining. She has published research papers in various National and International Journals, National and International Conferences. She also attended many National Seminars/FDP/Workshops Etc.



Dr Duvvada Rajeswara Rao Professor PhD. He had published research papers at National and International Journals, National and International Conferences. At present he is working as Professor at KL University Vijayawada.