SMDS: Secure Model for Cloud Data Storage

Krunal Suthar
Patel Institute of Technology
Ratibad, Bhopal

Parmalik Kumar
Patel Institute of Technology
Ratibad, Bhopal

Hitesh Gupta
Patel Institute of Technology
Ratibad, Bhopal

ABSTRACT

Cloud computing is a computing technique, where a large group of systems are connected to private or public networks, where data owner can store his data on remote systems and frees himself from storage burden and uses the data on-demand, anytime, everywhere. As, a Cloud data user does not possess direct control of his data, security is one of the few challenging issues which needs to be addressed. Security in Cloud computing can be addressed in many directions viz. authentication, integrity, confidentiality and many more. Data integrity or correctness is an issue where there may be some unauthorized alteration in the data without consent of the data owner. In this paper we address the issue of storage correctness in Cloud computing and propose operational algorithms which may be used to build a complete solution.

General Terms

Algorithms, Verification

Keywords

Cloud Computing, Data Storage Correctness, Security, Confidentiality.

1. INTRODUCTION

CLOUD computing is an approach where software, hardware and/or other resources are provisioned "as a Service". Cloud brings some benefits to its users such as relief from the burden of storage management, universal access to data, ubiquitous, lower capital expenditure etc. Various issues related to Cloud computing includes Security of data from theft, Data Integrity on Cloud, Secure transmission of data to and from Cloud sever, Verifying files without much overhead/Computation , rights management, maintain security during sharing and many more.

Data storage correctness or some time more generally referred as data integrity verification is one of chief Cloud security problems. Data can be altered by unauthorized entity without intimating to data owner. How would the data owner make sure that his data has not been modified by other intruders (or may be by the Cloud provider itself, accidently or intentionally). So detecting such kind of unlawful activities on data is an utmost priority issue. Data storage correctness schemes can be classified into two categories (a) Without Trusted Third Party (TTP) and (b) with TTP, based on who makes the verification. In case of TTP, an extra Third Party Auditor (TPA), some time in form of extra hardware or cryptographic coprocessor is used. This hardware scheme provides better performance due to dedicated hardware for the auditing process but has some drawbacks such as single TTP resulting into bottleneck in the system, mutually agreeing on a common TTP where there are thousands of users across the globe. Due to such kind of reasons, we prefer an approach where the functionalities of TPA is integrated in form of client application and the application can be downloaded by cloud user from cloud server. This client application provides all the cryptographic functionalities to achieve the goals of integrity, authentication and confidentiality. As this is a software approach, the performance of the overall system may not be comparable to dedicated hardware kind of TTP alternatives. To improve performance, we emphasize offline execution of computationally costly cryptographic algorithms.

As in traditional network security, we try to protect the confidentiality of data in its two stages of data life cycle viz. data at rest and data in transition. For data at rest, symmetric key encryption techniques (E.g. AES, TDES, DES etc.) are recommended, which are secure but more time consuming approaches. For data in transition, we recommend SSL kind of already available secure protocols. For integrity verification, we rely on hash functions such as SHA-1, MD-5.The rest of the paper is organized as follows. Section II gives details about related work. Section III explains the recommended operational details to achieve the goals. We analyze the approach in Section IV. Section V contains conclusion and future work followed by list of references used at the end.

2. RELATED WORK

Authors of [1] recommend a design of cryptographic cloud data storage and suggest various cryptographic approaches to achieve access control, authentication, confidentiality and non-repudiation. Authors of [2] propose a Data Storage Security Model provides storage correctness without affecting Cloud's dynamic nature while maintaining communication cost between Cloud server and users. User can verify their data with very less overhead. It also provides a flexible security option based on sensitivity of the data defined by the data owner.

Researchers of [3] [4] address problem of access control mechanism using cryptographic techniques which degrades the performance and increase computation cost for management of key at user as well Cloud server side. They give solution by capability based access control scheme which gives surety that only valid user having rights to access data available on Cloud. They also propose and modified version of Diffie-Hellman key exchange scheme for sharing symmetric key securely. Motivated by these papers, we are proposing operational algorithms for achieving the above mentioned goals.

3. ALGORITHM FOR OPERATIONAL STEPS

There are mainly three active components in the system:

(i) Data Owner (DO), who stores data on Cloud and can allow accessing of its data to other Cloud users.

(ii) Data requesters (DR), who use the data based on credentials received from the cloud owner.

(iii) Cloud server (CS) is a central component that provides storage as a service and works as a bridge between Data Owner and Data requester.

Table 1: Notation

PK	Public Key
PR	Private Key
Enc	Encryption
SK	Symmetric Key
DO	Data Owner
DR	Data requester
CS	Cloud Server
CHash	Hash code at Client
SHash	Hash at server
DO_ID	Data Owner ID
DR_ID	Data requester ID

We wish to achieve following goals. First, Cloud server neither should learn any information from Cloud users' data nor should misuse the same. Second, we also wish to offer an option to Cloud user for selecting encryption option for their data. Third, aim to achieve light weight integrity verification process for checking unauthorized change in original data, without requiring local copy of the data. Fourth, secure key management. Fifth, flexible access rights management.

The figure1 shows sequence of operation performed. The overall function is divided into 9 different phases. The details of phases are given below.

During Phase 1 the DO/DR generate a key pair using an public key encryption scheme in single Step which is used for encrypt the data during transmission.

Step 1: DO/DR Generate Key pair.

Aim of Phase2 is to get ID from Cloud server and send Registration detail on Cloud server. We achieve same using four steps. In First step, DO/DU sends the public key to the Cloud server. Cloud server Generate ID store key and ID, and send ID back to User in step 2. In step 3 users send registration details like Password, Mail id etc. and in last Step Cloud server store information in database and send confirmation.

Step 1: Send {PK}
Step 2: CS Generate and send {ID} back to DO/DR
Step 3: DO/DR send {ID, password, email etc.}
Step 4: CS store information and send Confirmation.

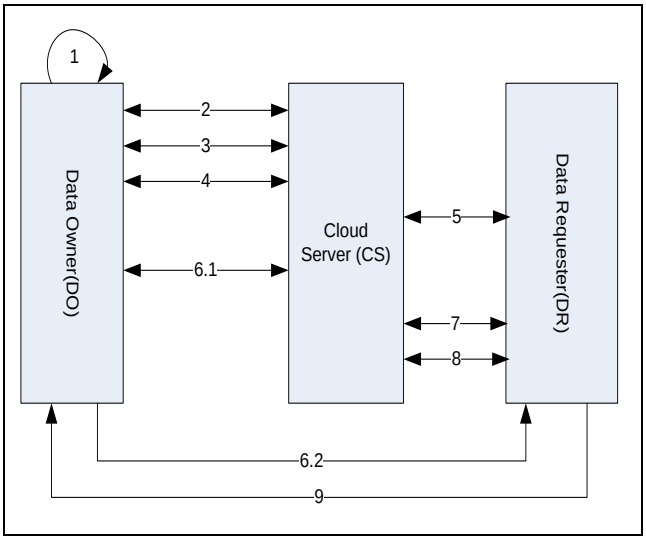

Figure 1: Sequence of operation

Aim of Phase 3 is to generate data locally and send it on Cloud server. In step1 Data owner encrypt the file using Symmetric key which is generated by different available encryption algorithm. During step2 Hash code calculate from encrypted file which is produce in step1 and store in database. Owner again encrypt file using its private key and the newly encrypted file with some other information sent on Cloud in step3. During last step Cloud server stores the file and makes necessary change in database and sends confirmation to Owner.

Step 1: Enc_fl <- Enc_{SK}(File))
Step 2: CHash <- Hash(($_{SHA\text{-}256/MD5}$(enc_fl)))
Step 3: Enc_Auth_file <- Enc_{PR}(enc_file))
Send {Enc_Auth_file}
Step 4: CS store {Enc_Auth_file}, update D/B.

During phase 4 our aim is to check integrity of User's data on Cloud server. This can achieve in three Steps. During first step Data owner send its id, and based on id Cloud server Reply with list of files which is owned by Owner. In step 2 Owner select a file from list and selected file name send to Cloud server. After some authentication Cloud server calculate hash code and send hash back to user in step3. User compares Hash locally after decrypting in last step.

Step 1: Send {DO_ID}
Generate{Fl_lst}, and Send{Fl_lst}
Step 2: Send {File name}
Step 3: Calculate Hash, and Send {Enc{SHash}}
Step 4: Compare {CHash, SHash}

Aim of phase 5 is to send request for rights. During first step Requester sends the ID of Owner to the Cloud user from whom it wants file access. In step2 Cloud server generate file list from id and send file list back to Requester. Requester select file from list and send selection with request rights (i.e. Read, Write) on Cloud server in Step3. During last step Cloud server make status of request as pending.

```
Step 1:  Send {DO_ID}
Step 2:  Fl_lst <- Generate file list from {DO_ID}
         Send{Fl_lst}
Step 3:  Retrieve (fl_lst)
         Send {File name(fl_lst), Rights(R/W)}
Step 4:  Make request status 'Pending'
```

During Phase 6 our wish is to either granting or denying rights request generated in phase5 and send Symmetric key in case of granting. This is achieving in four steps. During first step Owner send its id on Cloud server and gets list of pending rights for its owned file from Cloud server. In step2 Owner grants or denies the request and status sends to Cloud server, who makes changes in database accordingly. In step 3 Owner send a Symmetric key for the file to Requester in case of granting the rights.

```
Step 1: Send{DO_ID}
        Receives {file list}
Step 2: DO Send{ File,(Grant/Deny) }
Step 3: Send {SK} to DR
```

Aim of phase 7 is to downloading file from Cloud server. Here the Data requester sends its id on Cloud server and receives list of only those files for which its request are granted by Owner. During step 2 requester select file and send selected file name to Cloud. Cloud server send file contents to requester in step 3.

```
Step 1: Send {DR_ID}
        Receives {file list}
Step 2: Send {File name}
Step 3: Send File data.
```

During phase 8 our wish is to handle data which is modified by requester. The procedure for first three phase are similar to phase3 except the step 4 in which requester rights for file modification are checked by Cloud server before overwriting the file on Cloud storage.

```
Step 1:  Enc_fl <- Enc(SK(File))
Step 2:  CHash <- Hash((SHA-256/MD5(enc_fl)))
Step 3:  Enc_Auth_file <- Enc(PR(enc_file))
         Send { Enc_Auth_file }
Step 4:  CS Check for rights
         If (true)
              Accept Request and update D/B
         Else
              Deny modification
```

During last phase our aim is to provide hash of modified file to Owner. The requester sends hash code to Owner and after some verification owner update hash in its local database.

```
Step 1:  Send{hash}
Step 2:  Retrieve and Update Database.
```

4. ANALYSIS OF SCHEME

Following paragraph illustrates the security and general analysis of the system and how we achieve the goals mentioned earlier.

1) *Data Confidentiality :* As we store the data in encrypted form on cloud, and keep the keys and the algorithm itself, unknown from Cloud server, it is next to impossible for the server to either learn the data or to misuse them.
2) *Security options* : As all the data to be stored on Cloud may not be highly sensitive, we take inputs from the data owner himself, and accordingly select cryptographic algorithms for him.
3) *Lightweight Verification*: For integrity verification, cloud user/requestor can send request (in form of challenge) to cloud server for computing and submitting hash code of his encrypted file. Upon checking some validations, cloud server computes a hash code of the file and returns the same to the requestor (in form of response). The size of this code is very small (in terms of few bytes) which reduces communication overhead. Also note that, computing the hash code is an offline function at cloud server side. In this way, we save computation plus communication time, hence improve performance.
4) *Key Management* : We have used the hybrid approach of using a combination of symmetric and asymmetric key encryption. Data encryption is done in a symmetric way and the key used for it is transferred to the data requester in an asymmetric way. Hence, utilizing secure approach for data encryption and fast operation for key transfer is adopted.
5) *Access Rights*: Access rights can be granted from data owner to data requester with the help of small SQL grant operations. In case of revoking a grant, again the same kind of SQL revoke statement can be used. Important, thing here to mention is, in case of granting operation, data owner may be talking to data requester, but in case of revoking the rights, it will issue instructions directly to Cloud server, of course through SQL statement. Hence, it is quite a simple operation.
6) *No data duplication***:** Without asking local copy of data, correctness can be measured even data is in encrypted form. The decryption is also done offline at the site of DO/DR. Here data are not moving from one position to another in unencrypted format.

This next part gives technical analysis of model. Here we compare our security model with the already available Cloud's provider. Table 2 shown an comparison based on various already available storage security approaches which works based on cryptographic without involvement of trusted third party. Those approaches do not handle security issue in all the directions. For example Venus [19], and cloudZone [22] do not address issue of data privacy and confidentiality whereas EPPS [20] and CloudSeal[21] do not address the problem of integrity verification. Our model SMDS handles the problem of integrity as well data confidentiality. Other than this, SMDS offers flexible security options to user based on sensitivity of their data which is not provided by any of the provider except EPPS. Secure access control, managing access rights, audit trail, high performance and less overhead are some other benefits offered by SMDS.

Table 2: Technical comparison

Criteria Group →	Integrity oriented measures		Confidentiality oriented measures				Others						
Individual Criteria → / Cloud Providers ↓	Public verifiability	Verification cost	Encryption (Data at rest)	Encryption (Data in transition)	Access Control Mechanism	Flexible security options	Access rights mechanism	Permanently deletion of data	Versioning	Audit Trail	Scalability	Performance	Overhead
CouldSeal	X	X	√	√	√	X	√	X	X	X	X	↓	↑
CloudZone	X	√	X	X	X	X	√	X	X	X	X	↑	↓
Venus	X	√	X	X	√	X	√	X	X	X	√	↑	↓
EPPS	X	X	√	√	X	√	X	X	X	X	X	↑	↓
SMDS	X	√	√	√	√	√	√	X	X	√	√	↑	↓

5. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed data storage correctness issue of Cloud computing. We have provided some operational algorithms to offer an effective flexible security options using Modern symmetric encryption schemes which provide confidentiality based on sensitivity of user's data as well integrity verification with low computation cost on Client. This feature may be useful for thin Client who may do not having much processing power. Authentication, access control, non-duplication of original data on Cloud Etc. are few of the other security goals of the proposal. Symmetric key sharing is handled with public key cryptography, to achieve faster performance and low computational overhead. In Last Analysis part we compare the proposed model with some available approaches and from the analysis we can say the proposed model provides almost all the features which is required to make an complete solution. Our future target is to implement such a security suit which provides all the goals mentioned.

6. REFERENCES

[1] Kamara, S., Lauter, K.: "Cryptographic cloud storage". In: *Proceedings of the 14th international conference on Financial cryptography and data security*, FC'10, pp. 136-149. Springer-Verlag, Berlin, Heidelberg (2010)

[2] Hiren B. Patel, Dhiren R. Patel, Bhavesh Borsania, Avi Patel "Data Storage Security Model for Cloud Computing" In *CNC-2012.*

[3] C. Hota, S. Sanka, M. Rajarajan, S. Nair, "Capability-based Cryptographic Data Access Control in Cloud Computing", in *International Journal of Advanced Networking and Applications*, Volume 01, Issue 01, 2011.

[4] Balakrishnan. S, Saranya. G, Shobana. S, Karthikeyan.S, "Introducing Effective Third Party Auditing (TPA) for Data Storage Security in Cloud" *In International Journal of Computer Science and Technology*, Vol. 2, Issue 2, June 2011.

[5] S. Sanka, C. Hota, M .Rajarajan, "Secure Data Access in Cloud Computing" *in IEEE Conference*, 2010.

[6] K. Kajendran, J. Jeyaseelan, J. Joshi, "An Approach for secures Data storage using Cloud Computing" *In International Journal of Computer Trends and Technology*- May to June Issue 2011

[7] S. Kumar, A. Saxena, "Data Integrity Proofs in Cloud Storage", in *IEEE Conference*, 2011.

[8] W. Luo, G. Bai, "Ensuring the Data Integrity In Cloud computing" In *Proceedings of IEEE CCIS*, 2011.

[9] C. Wang, Q. Wang, K.Ren, W. Lou, "Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing", in *IEEE INFOCOM 2010*, San Diego, CA, March 2010.

[10] C. Wang, Q. Wang, K. Ren, W. Lou, "Ensuring Data Storage Security in Cloud Computing".

[11] Q. Wang, C. Wang, K. Ren, W. Lou, J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing", *14th European Symposium on Research in Computer Security* (ESORICS'09), 2009.

[12] Chuang, I.H., Li, S.H., Huang, K.C., Kuo, Y.H.: "An effective privacy protection scheme for cloud computing". In: *Advanced Communication Technology (ICACT),* 2011 13th International Conference on, pp. 260 -265 (2011).

[13] B. Gowrigolla, S. Sivaji, M. Masillamani, "Design and Auditing of Cloud Computing Security", *in IEEE Conference,* 2010.

[14] Betty Huang "Analysis of the RSA Encryption Algorithm" June 16, 2010

[15] Advanced encryption standard (aes) (fips pub 197) (2001).

[16] FIPS 46-3: Data Encryption Standard (DES). (fips pub 46- 3) (1999).

[17] Rivest, R., Shamir, A., Adleman, L.: "A method for obtaining digital signatures and public-key cryptosystems" *Communications of the ACM 21*, 120-126 (1978).

[18] Goyal, V., Pandey, O., Sahai, A., Waters, B.: "Attribute-based encryption for fine-grained access control of encrypted data" In *Proceedings of the 13th ACM conference on Computer and communications security CCS* 06 p. 89 (2006)

[19] Alexander Shraer "Venus verification for untrusted Cloud storage" CCSW '10 Proceedings of the 2010 ACM workshop on Cloud computing security workshop.2010.

[20] I-Hsun Chuang Syuan-Hao Li ; Kuan-Chieh Huang ; Yau-Hwang Kuo "An effective privacy protection scheme for cloud computing" In 13th International Conference on Advanced Communication Technology, 20122, pp. 260-265.

[21] Huijun Xiong, Xinwen Zhang, Danfeng Yao, Xiaoxin Wu, Yonggang Wen, "Towards End-to-End Secure Content Storage and Delivery with Public Cloud" Proceedings of the second ACM conference on Data and Application Security and Privacy, pp 257-266,2012.

[22] Ostrovsky, R., Sahai, A., Waters, B.: "Attribute-based encryption with non-monotonic access structures" In: *ACM CCCS* (2007)

[23] Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: "Provable data possession at untrusted stores". In*: Proceedings of the ACM conference on Computer and Communications security, CCS '07,* pp. 598– 609. ACM, NY. (2007)