Replicated Data Integrity Verification in Cloud

Raghul Mukundan
Department of Computer Science
Missouri University of Science and Technology
rmgq8@mst.edu

Sanjay Madria
Department of Computer Science
Missouri University of Science and Technology
madrias@mst.edu

Mark Linderman
Air Force Research Lab
Rome, NY
mark.linderman@rl.af.mil

Abstract

Cloud computing is an emerging model in which computing infrastructure resources are provided as a service over the Internet. Data owners can outsource their data by remotely storing them in the cloud and enjoy on-demand high quality applications and services from a shared pool of configurable computing resources. However, since data owners and cloud servers are not in the same trusted domain, the outsourced data may be at risk as the cloud server may no longer be fully trusted. Therefore, data integrity is of critical importance in such a scenario. Cloud should let either the owners or a trusted third party to audit their data storage without demanding a local copy of the data from owners. Replicating data on cloud servers across multiple data centers provides a higher level of scalability, availability, and durability. When the data owners ask the Cloud Service Provider (CSP) to replicate data at different servers, they are charged a higher fee by the CSP. Therefore, the data owners need to be strongly convinced that the CSP is storing all the data copies that are agreed upon in the service level contract, and the data-update requests issued by the customers have been correctly executed on all the remotely stored copies. To deal with such problems, previous multi copy verification schemes either focused on static files or incurred huge update costs in a dynamic file scenario. In this paper, we propose some ideas under a Dynamic Multi-Replica Provable Data Possession scheme (DMR-PDP) that prevents the CSP from cheating; for example, by maintaining fewer copies than paid for. DMR-PDP also supports efficient dynamic operations like block modification, insertion and deletion on data replicas over cloud servers.

1 Introduction

When users store data in the cloud, their main concern is whether the cloud can maintain the data integrity and data can be recovered when there is a data loss or server failure. Cloud service providers (CSP), in order to save storage cost, may tend to discard some data or data copies that is not accessed often, or mitigate data to second-level storage devices. CSPs may also conceal data loss due to management faults, hardware

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

This publication has been cleared for public release, distribution unlimited by AFRL (case number 88ABW-2012-6360).

failures or attacks. Therefore, a critical issue in storing data at untrusted CSPs is periodically verifying whether the storage servers maintain data integrity; store data completely and correctly as stated in the service level agreement (SLA).

Data replication is a commonly used technique to increase the data availability in cloud computing. Cloud replicates the data and stores them strategically on multiple servers located at various geographic locations. Since the replicated copies look exactly similar, it is difficult to verify whether the cloud really stores multiple copies of the data. Cloud can easily cheat the owner by storing only one copy of the data. Thus, the owner would like to verify at regular intervals whether the cloud indeed possesses multiple copies of the data as claimed in the SLA. In general, cloud has the capability to generate multiple replicas when a data owner challenges the CSP to prove that it possesses multiple copies of the data. Also, it is a valid assumption that the owner of the data may not have a copy of the data stored locally. So, the major task of the owner is not only to verify that the data is intact but also to recover the data if any deletions/corruptions of data are identified. If the data owner during his verification using DMR-PDP scheme detects some data loss in any of the replicas in the cloud, he can recover the data from other replicas that are stored intact. Since, the replicas are to be stored at diverse geographic locations, it is safe to assume that a data loss will not occur at all the replicas at the same time.

Provable data possession (PDP) [2] is a technique to audit and validate the integrity of data stored on remote servers. In a typical PDP model, the data owner generates metadata/tag for a data file to be used later for integrity verification. To ensure security, the data owner encrypts the file and generates tags on the encrypted file. The data owner sends the encrypted file and the tags to the cloud, and deletes the local copy of the file. When the data owner wishes to verify the data integrity, he generates a challenge vector and sends it to the cloud. The cloud replies by computing a response on the data and sends it to the verifier/data owner to prove that multiple copies of the data file are stored in the cloud. Different variations of PDP schemes such as [2], [4], [6], [7], [9], [10], [11], [12], [15] were proposed under different cryptographic assumptions. But most of these schemes deal only with static data files and are valid only for verifying a single copy. A few other schemes such as [3], [5], [8], [13], [14] provide dynamic scalability of a single copy of a data file for various applications which mean that the remotely stored data can not only be accessed by the authorized users, but also be updated and scaled by the data owner.

In this paper, we propose a scheme that allows the data owner to securely ensure that the CSP stores multiple replicas. A simple way to make the replicas look unique and differentiable is using probabilistic encryption schemes. Probabilistic encryption creates different cipher texts each time the same message is encrypted using the same key. Thus, our scheme uses homomorphic probabilistic encryption to create distinct replicas/copies of the data file and BLS signatures [17] to create constant amount of metadata for any number of replicas. Probabilistic encryption encrypts all the replicas with the same key. Therefore, in our scheme the data owner will have to share just one decryption key with the authorized users and need not worry about CSP granting access to any of the replicas to the authorized users. The homomorphic property of the encryption scheme helps in efficient file updates. The data owner has to encrypt the difference between the updated file and the old file and send it to the cloud, which updates all the replicas by performing homomorphic addition on the file copies. Any authenticated data structure, e.g, Merklee Hash Trees and Skiplist can be used with our scheme to ensure that the cloud uses the right file blocks for data integrity verification. However, the ways to efficiently manage authenticated data structures in the cloud is not within the scope of our paper.

Organization: The rest of the paper is organized as follows. Overview of the related work is provided in Section 2 followed by the problem definition in Section 3, and a detailed description of our scheme in Section 4. Future work is discussed in Section 5.

2 Related Work

Ateniese et al. [2] were the first to define the Provable Data Possession (PDP) model for ensuring the possession of files on untrusted storages. They made use of RSA-based homomorphic tags for auditing outsourced data. However, dynamic data storage and multiple replica system are not considered in this scheme. In their subsequent work [12], they proposed a dynamic version which supports very basic block operations with limited functionality, and does not support block insertions. In [13], Wang et al. considered dynamic data storage in a distributed scenario, and proposed a challenge-response protocol which can determine data correctness as well as locate possible errors. Similar to [12], only partial support for dynamic data operation is considered. Erway et al. [14] extended the PDP model in [2] to support provable updates to stored data files using rank-based authenticated skip lists. However, efficiency of their scheme remains unclear and these schemes hold good only for verifying a single copy.

Curtmola et.al [1] proposed Multiple-Replica PDP (MR-PDP) scheme wherein the data owner can verify that several copies of a file are stored by a storage service provider. In their scheme, distinct replicas are created by first encrypting the data and then masking it with randomness generated from a Pseudo-Random Function (PRF). The randomized data is then stored across multiple servers. The scheme uses RSA signatures for creation of tags. But, their scheme did not address how the authorized users of the data can access the file copies from the cloud servers noting that the internal operations of the CSP are opaque and do not support dynamic data operations. Ayad F. Barsoum et al. [16] proposed creation of distinct copies by appending replica number to the file blocks and encrypting it using an encryption scheme that has strong diffusion property, e.g., AES. Their scheme supports dynamic data operations but during file updates, the copies in all the servers should be encrypted again and updated on the cloud. This scheme suits perfectly for static multiple replicas but proves costly in a dynamic scenario. BLS signatures are used for creating tags and authenticated data structures like Merklee Hash Trees are used to ensure right file blocks are used during verification. Authorized users of the data should know random numbers in [1] and replica number in [16] to generate the original file.

3 Dynamic Multi-Replica Provable Data Possession (DMR-PDP) Scheme

The cloud computing model considered in this work consists of three main components as illustrated in Figure 1: (i) a data owner that can be an individual or an organization originally possessing sensitive data to be stored in the cloud; (ii) a CSP who manages cloud servers and provides paid storage space on its infrastructure to store the owner's files and (iii) authorized users - a set of owner's clients who have the right to access the remote data and share some keys with the data owner.

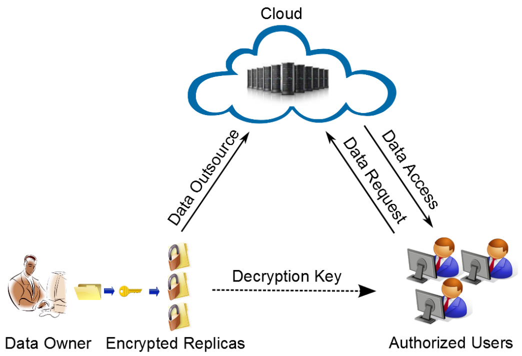


Figure 1: Cloud Computing Data Storage Model.

3.1 Problem Definition and Design Goals

More recently, many data owners relieve their burden of local data storage and maintenance by outsourcing their data to a CSP. CSP undertakes the data replication task in order to increase the data availability, durability and reliability but the customers have to pay for using the CSPs storage infrastructure. On the other hand, cloud customers should be convinced that the (1) CSP actually possesses all the data copies as agreed upon, (2) integrity of these data copies are maintained, and (3) the customers are receiving the service that they are paying for. Therefore, in this paper, we address the problem of securely and efficiently creating multiple replicas of the data file of the owner to be stored over untrusted CSP and then auditing all these copies to verify their completeness and correctness. Our design goals are summarized below:

1. Dynamic Multi-Replica Provable Data Possession (DMR-PDP) protocols should efficiently and securely provide the owner with strong evidence that the CSP is in possession of all the data copies as agreed upon and that these copies are intact.

2. Allowing the users authorized by the data owner to seamlessly access a file copy from the CSP.

3. Using only a single set of metadata/tags for all the file replicas for verification purposes.

4. Allowing dynamic data operation support to enable the data owner to perform block-level operations on the data files while maintaining the same level of data correctness assurance.

5. Enabling both probabilistic and deterministic verification guarantees.

3.2 Preliminaries and Notations

In this section, we provide details of the Bilinear mapping and Paillier Encryption schemes used in our present work.

1. Assume that F, a data file to be outsourced, is composed of a sequence of *m* blocks, i.e., F = {b_1, b_2,..,b_m}.

2. F_i = {b_{i1}, b_{i2},....,b_{im}} represents the file copy *i*.

3. **Bilinear Map/Pairing:** Let G_1, G_2, and G_T be cyclic groups of prime order *a*. Let *u* and *v* be generators of G_1 and G_2, respectively. A bilinear pairing is a map e : G_1 x G_2 $\rightarrow$ G_T with the following properties:

   - Bilinear: $e(u_1u_2, v_1) = e(u_1, v_1) . e(u_2, v_1)$, $e(u_1, v_1v_2) = e(u_1, v_1) . e(u_1,v_2)$ $\forall$ $u_1, u_2 \in G_1$ and $v_1, v_2 \in G_2$
   - Non-degenerate: $e(u, v) \neq 1$
   - There exists an efficient algorithm for computing *e*
   - $e({u_1}^x, {v_1}^y) = e(u_1, v_1)^{xy}$ $\forall$ $u_1 \in G_1$; $v_1 \in G_2$, and x, y $\in Z_a$

4. H(.) is a map-to-point hash function: $\{0, 1\}^* \rightarrow G_1$.

5. **Homomorphic Encryption:** A homomorphic encryption scheme has the following properties.

   - $E(m_1 + m_2) = E(m_1) +_h E(m_2)$ where $+_h$ is a homomorphic addition operation.
   - $E(k*m) = E(m)^k$.

   where E(.) represents a homomorphic encryption scheme and m, m_1, m_2 are messages that are encrypted and k is some random number.

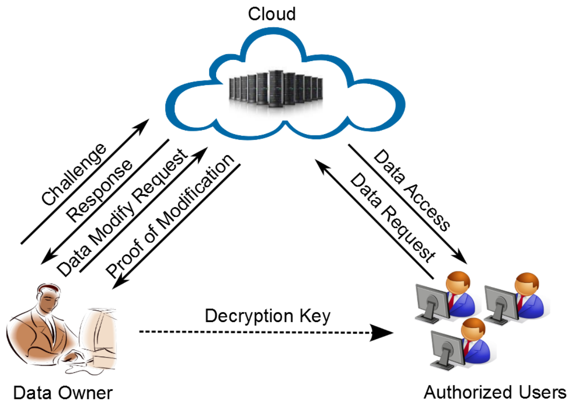


Figure 2: DMR-PDP Scheme

6. **Paillier Encryption:** Paillier cryptosystem is a homomorphic probabilistic encryption scheme. The steps are as follows.
    - Compute N = p * q and λ = LCM (p-1, q-1), where p, q are two prime numbers.
    - Select a random number g such that its order is a multiple of N and $g \in Z_N{}^{2*}$.
    - Public key is (N, g) and secret key is λ, where N = p*q.
    - Cipher text for a message m is computed as c = g^m r^N mod N^2 where r is a random number and r $\in Z_N{}^*$, c $\in Z_N{}^{2*}$ and $m \in Z_N$.
    - Plain text is obtained by m = L(c^λ mod N^2) * (L(g^λ mod N^2))$^{-1}$ mod N.
7. Properties of public key g in Paillier Scheme
    - $g \in Z_N{}^{2*}$.
    - If g = (1 + N) mod N^2, it has few interesting properties
        (a) Order of the value (1 + N) is N.
        (b) $(1 + N)^m \equiv (1 + mN) \bmod N^2$. (1 + mN) can be used directly instead of calculating $(1 + N)^m$. This avoids the costly exponential operation during data encryption.

3.3 DMR-PDP Construction

In our approach, the data owner creates multiple encrypted replicas and uploads them on to the cloud. The CSP stores them on one or multiple servers located at various geographic locations. The data owner shares the decryption key with a set of authorized users. In order to access the data, an authorized user sends a data request to the CSP and receives a data copy in an encrypted form that can be decrypted using a secret key shared with the owner. The proposed scheme consists of seven algorithms: KeyGen, ReplicaGen, TagGen, Prove, Verify, PrepareUpdate and ExecUpdate. The overview of the communication involved in our scheme is shown in 2.

1. (pk, sk) $\leftarrow$ KeyGen(). This algorithm is run by the data owner to generate a public key *pk* and a private key *sk*. The data owner generates three sets of keys.
    (a) Keys for data tags : This key is used for generating tags for the data. The data owner selects a bilinear map e and selects a private key l $\in Z_a$. Public key is calculated as y = $v^l \in G_2$.

(b) Keys for data : This key is used for encrypting the data and thereby creating multiple data copies. The data owner selects paillier public keys (N, g) with g = (1 + N) mod N^2 and secret key λ.

(c) PRF key : The data owner generates a PRF key Key_{PRF} which generates *s* numbers. These *s* numbers are used in creating *s* copies of the data. Each number is used in creating one data copy. Let $\{k_1, k_2,..,k_s\} \in Z_N{}^*$ be the numbers generated by the PRF key. Key_{PRF} is maintained confidentially by the data owner and hence the *s* numbers used in creating multiple copies are not known to the cloud.

2. $\{F_i\}_{1 \leq i \leq s} \leftarrow$ ReplicaGen (s, F). This algorithm is run by the data owner. It takes the number of replicas *s* and the file *F* as input and generates *s* unique differentiable copies $\{F_i\}_{1 \leq i \leq s}$. This algorithm is run only once. Unique copies of each file block of file F is created by encrypting it using a probabilistic encryption scheme, e.g., Paillier encryption scheme.

   Through probabilistic encryption, encrypting a file block *s* times yields *s* distinct cipher texts. For a file F = $\{b_1, b_2,..,b_m\}$ multiple data copies are generated using Paillier encryption scheme as F_i = $\{(1+N)^{b_1}(k_i r_{i1})^N, (1+N)^{b_2}(k_i r_{i2})^N,.., (1+N)^{b_m}(k_i r_{im})^N\}_{1 \leq i \leq m}$. Using Paillier's properties the above result can be rewritten as $F_i = \{(1+b_1N)(k_i r_{i1})^N, (1+b_2N)(k_i r_{i2})^N,.., (1+b_mN)(k_i r_{im})^N\}_{1 \leq i \leq m}$, where *i* represents the file copy number, k_i represents the numbers generated from PRF key Key_{PRF} and r_{ij} represents any random number used in Paillier encryption scheme. k_i is multiplied by a random number r_{ij} and the product is used for encryption. The presence of k_i in a file block identifies which file copy the file block belongs to. All these file copies yield the original file when decrypted . This allows the users authorized by the data owner to seamlessly access the file copy received from the CSP.

3. $\phi \leftarrow$ TagGen (sk, F). This algorithm is run by the data owner. It takes the private key sk and the file F as input and outputs the tags ϕ. We use BLS signature scheme to create tags on the data. BLS signatures are short and homomorphic in nature and allow concurrent data verification, which means multiple data blocks can be verified at the same time. In our scheme, tags are generated on each file block b_i as $\phi_i = (H(F) . u^{b_i N})^l \in G_1$ where $u \in G_1$ and $H(.) \in G_1$ represents hash value which uniquely represents the file F. The data owner sends the tag set $\phi = \{\phi_i\}_{1 \leq i \leq m}$ to the cloud.

4. *P* $\leftarrow$ Prove (F, ϕ, *challenge*). This algorithm is run by the CSP. It takes the file replicas of file F, the tags ϕ and *challenge* vector sent by the data owner as input and returns a proof *P* which guarantees that the CSP is actually storing *s* copies of the file F and all these copies are intact. The data owner uses the proof *P* to verify the data integrity. There are two phases in this algorithm:

   (a) **Challenge:** In this phase the data owner challenges the cloud to verify the integrity of all outsourced copies. There are two types of verification schemes:

      i. Deterministic - here all the file blocks from all the copies are used for verification.

      ii. Probabilistic - only a few blocks from all the copies are used for verification. A Pseudo Random Function key (PRF) is used to generate random indices ranging between 1 and *m*. The file blocks from these indices are used for verification. In each verification a percentage of the file is verified and it accounts for the verification of the entire file.

   At each challenge, the data owner chooses the type of verification scheme he wishes to use. If the owner chooses the deterministic verification scheme, he generates one PRF key, Key_1. If he chooses the probabilistic scheme he generates two PRF keys, Key_1 and Key_2. PRF keyed with Key_1 generates *c* ($1 \leq c \leq m$) random file indices which indicates the file blocks that CSP should use for verification. PRF keyed with Key_2 generates *s* random values and the CSP should use each of these random numbers for each file copy while computing the response. The data owner sends the generated keys to the CSP.

(b) **Response:** This phase is executed by the CSP when a challenge for data integrity verification is received from the data owner. Here, we show the proof for probabilistic verification scheme (the deterministic verification scheme also follows the same procedure). The CSP receives two PRF keys, Key$_1$ and Key$_2$ from the data owner. Using Key$_1$, CSP generates a set {C} with *c* (1$\leq$ c $\leq$ m) random file indices ({C} $\in$ {1, 2,..,m}), which indicate the file blocks that CSP should use for verification. Using Key$_2$, CSP generates 's' random values T = {t$_1$, t$_2$,..,t$_s$}. The cloud performs two operations. One on the tags and the other on the file blocks.

i. Operation on the tags: Cloud multiplies the file tags corresponding to the file indices generated by PRF key Key$_1$.

$$\begin{aligned}\sigma &= \prod_{j\in C}(H(F).u^{b_jN})^l \\ &= \prod_{j\in C}H(F)^l.\prod_{j\in C}u^{b_jNl} \\ &= H(F)^{cl}.u^{Nl\sum\limits_{j\in C}(b_j)}\end{aligned}$$

ii. Operation on the file blocks: The cloud first takes each file copy and multiplies all the file blocks corresponding to the file indices generated by PRF key Key$_1$. The product of each copy is raised to the power the random number generated for that copy by the PRF key Key$_2$. The result of the above operation for each file copy *i* is given by ($\prod\limits_{j\in C}(1+N)^{b_j}(k_ir_{ij})^N)^{t_i}$ mod N^2. The CSP then multiplies the result of each copy to get the result

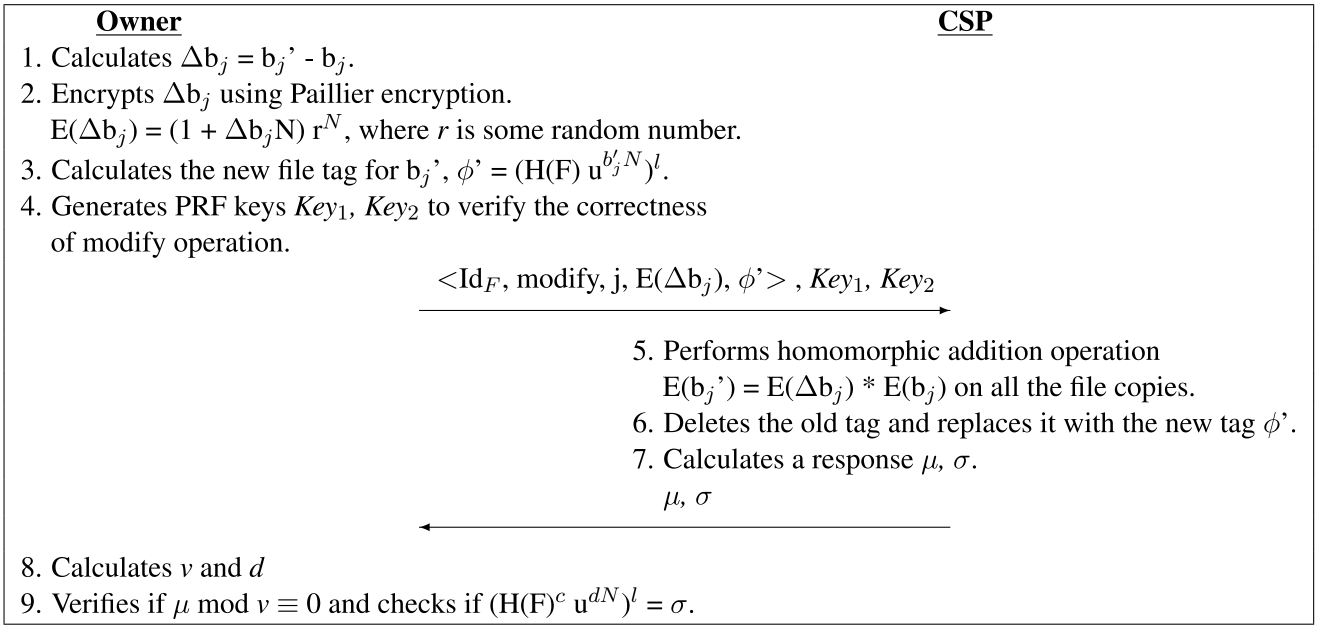
$$\begin{aligned}\mu &= \prod_{i=1}^{s}(\prod_{j\in C}(1+N)^{b_j}(k_ir_{ij})^N)^{t_i} \\ &= \prod_{i=1}^{s}(\prod_{j\in C}(1+N)^{b_jt_i}\prod_{j\in C}(k_ir_{ij})^{Nt_i}) \\ &= \prod_{i=1}^{s}((1+N)^{t_i\sum\limits_{j\in C}b_j}\prod_{j\in C}(k_ir_{ij})^{Nt_i}) \\ &= (\prod_{i=1}^{s}(1+N)^{t_i\sum\limits_{j\in C}b_j})(\prod_{i=1}^{s}((k_i)^{ct_iN}\prod_{j\in C}(r_{ij})^{Nt_i})) \\ &= ((1+N)^{\sum\limits_{i=1}^{s}t_i\sum\limits_{j\in C}b_j})(\prod_{i=1}^{s}(k_i)^{ct_iN})(\prod_{i=1}^{s}\prod_{j\in C}(r_{ij})^{Nt_i})\end{aligned}$$

Using properties of Paillier scheme, the above equation can be rewritten as

$$\mu = (1+N\sum_{i=1}^{s}(t_i)\sum_{j\in C}(b_j))(\prod_{i=1}^{s}(k_i)^{Nct_i})(\prod_{i=1}^{s}(\prod_{j\in C}(r_{ij})^{t_iN}))$$

The CSP sends σ and μ mod N^2 values to the data owner.

5. $\{1, 0\} \leftarrow$ Verify (pk, P). This algorithm is run by the data owner. It takes as input the public key pk and the proof P returned from the CSP, and outputs 1 if the integrity of all file copies is correctly verified or 0 otherwise. After receiving σ and μ values from the CSP, the data owner does the following:



Owner **CSP**

1. Calculates $\Delta b_j = b_j' - b_j$.
2. Encrypts Δb_j using Paillier encryption.
   $E(\Delta b_j) = (1 + \Delta b_j N)\, r^N$, where *r* is some random number.
3. Calculates the new file tag for b_j', $\phi' = (H(F)\, u^{b'_j N})^l$.
4. Generates PRF keys *Key*$_1$, *Key*$_2$ to verify the correctness of modify operation.

$$\xrightarrow{<Id_F,\ modify,\ j,\ E(\Delta b_j),\ \phi'>\ ,\ Key_1,\ Key_2}$$

5. Performs homomorphic addition operation
   $E(b_j') = E(\Delta b_j) * E(b_j)$ on all the file copies.
6. Deletes the old tag and replaces it with the new tag ϕ'.
7. Calculates a response μ, σ.

$$\xleftarrow{\mu,\ \sigma}$$

8. Calculates *v* and *d*
9. Verifies if $\mu \bmod v \equiv 0$ and checks if $(H(F)^c\, u^{dN})^l = \sigma$.

Figure 3: Block modification operation in the DMR-PDP scheme

(a) calculates $v = (\prod_{i=1}^{s} (k_i)^{t_i c N})$ and $d = Decrypt(\mu)/(\sum_{i=1}^{s} k_i)$. This can be calculated from the values generated from Key_{PRF} and *c*.

(b) checks if μ mod v $\equiv$ 0. This ensures that the cloud has used all the file copies while computing the response.

(c) checks if $(H(F)^c\, u^{dN})^l = \sigma$. This ensures that the CSP has used all the file blocks while computing the response. If options b and c are satisfied, it indicates that the data stored by the owner in the cloud is intact and the cloud has stored multiple copies of the data as agreed in the service level agreement.

6. *Update* ← PrepareUpdate (). This algorithm is run by the data owner to perform any operation on the outsourced file copies stored by the remote CSP. The output of this algorithm is an *Update* request. The data owner sends the *Update* request to the cloud and will be of the form <Id$_F$, BlockOp, j, b$_i$', ϕ'>, where Id$_F$ is the file identifier, BlockOp corresponds to block operation, j denotes the index of the file block, b$_i$' represents the updated file blocks and ϕ' is the updated tag. BlockOp can be data modification, insertion or delete operation.

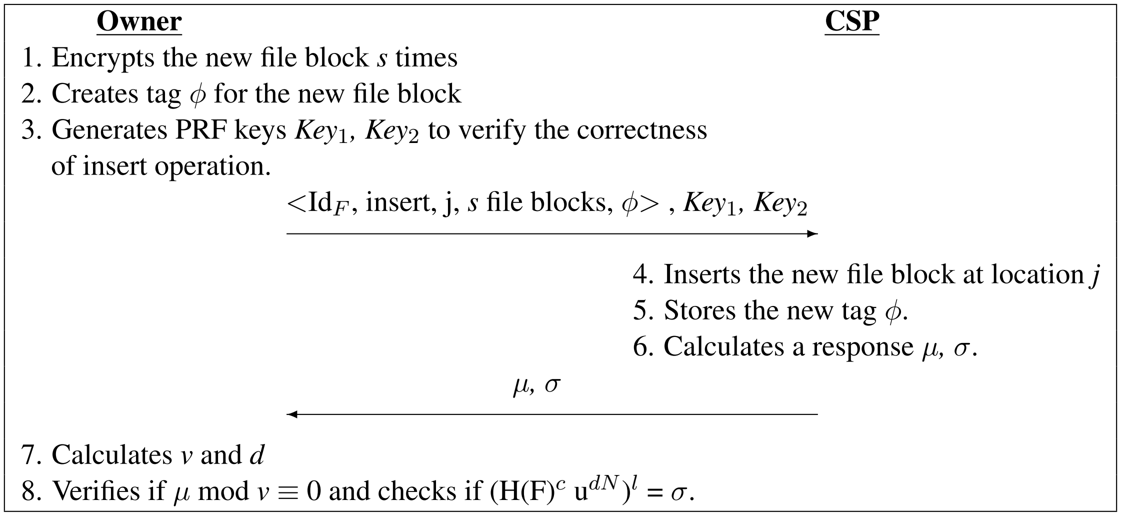
7. (F', ϕ') ← ExecUpdate (F, ϕ, *Update*). This algorithm is run by the CSP where the input parameters are the file copies F, the tags ϕ, and *Update* request (sent from the owner). It outputs an updated version of all the file copies F' along with updated signatures ϕ'. After any block operation, the data owner runs the challenge protocol to ensure that the cloud has executed the operations correctly. The operation in *Update* request can be modifying a file block, inserting a new file block or deleting a file block.

   (a) **Modification:** Data modification is one of the most frequently used dynamic operations. The data modification operation in DMR-PDP scheme is shown in Figure 3.

   (b) **Insertion:** In the block insertion operation, the owner inserts a new block after position *j* in a file. If the file *F* had *m* blocks initially, the file will have *m+1* blocks after the insert operation.

The file block insertion operation is shown in Figure 4.

(c) **Deletion:** Block deletion operation is the opposite of the insertion operation. When one block is deleted, indices of all subsequent blocks are moved one step forward. To delete a specific data block at position *j* from all copies, the owner sends a delete request <Id$_F$, delete, j, null, null> to the cloud. Upon receiving the request, the cloud deletes the tag and the file block at index *j* in all the file copies.



Owner

CSP

1. Encrypts the new file block *s* times
2. Creates tag ϕ for the new file block
3. Generates PRF keys Key_1, Key_2 to verify the correctness of insert operation.

<Id$_F$, insert, j, *s* file blocks, ϕ> , Key_1, Key_2

4. Inserts the new file block at location *j*
5. Stores the new tag ϕ.
6. Calculates a response μ, σ.

μ, σ

7. Calculates *v* and *d*
8. Verifies if μ mod $v \equiv 0$ and checks if $(\mathrm{H(F)}^c\, \mathrm{u}^{dN})^l = \sigma$.

Figure 4: Block insertion operation in the DMR-PDP scheme

4 Conclusions and Future Work

In this paper, we have discussed work related to the replicated data integrity preservation in a cloud environment and presented a Dynamic Multi-Replica Provable Data Possession scheme (DMR-PDP) to periodically verify the correctness and completeness of multiple data copies stored in the cloud. Our scheme also supports dynamic data update operations. All the data copies can be decrypted using a single decryption key, thus providing a seamless access to the data's authorized users. This scheme can be extended for multiple versions where only deltas can be stored in the cloud and owner can save on storage cost. Currently, we are implementing the proposed scheme for evaluating it in a real cloud platform using different performance metrics and comparing it with some of the existing methods. We also plan to extend this scheme for secure multi-version data where only one original and multiple deltas can be stored in the cloud.

References

[1] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," in 28th IEEE ICDCS, 2008, pp. 411-420.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in CCS '07: Proceedings of the 14th ACM Conference on Computer and Communications Security, New York, NY, USA, 2007, pp. 598-609.

[3] G. Ateniese, R. D. Pietro, L. V. Mancin, and G. Tsudik, "Scalable and efficient provable data possession," in SecureComm âĂŹ08: Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks, New York, NY, USA,2008, pp. 1-10.

[4] Y. Deswarte, J.-J. Quisquater, and A. SaÄśdane, "Remote integrity checking," in 6th Working Conference on Integrity and Internal Control in Information Systems (IICIS), S. J. L. Strous, Ed., 2003, pp. 1-11.

[5] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in CCS âĂŹ09: Proceedings of the 16th ACM Conference on Computer and Communications Security, New York, NY, USA, 2009, pp. 213-222.

[6] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," Cryptology ePrint Archive, Report 2006/150, 2006.

[7] P. Golle, S. Jarecki, and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity," in FC'02: Proceedings of the 6th International Conference on Financial Cryptography, Berlin, Heidelberg, 2003, pp. 120-135.

[8] Z. Hao, S. Zhong, and N. Yu, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability," IEEE Transactions on Knowledge and Data Engineering, vol. 99, no. PrePrints, 2011.

[9] E. Mykletun, M. Narasimha, and G. Tsudik, "Authentication and integrity in outsourced databases," Trans. Storage, vol. 2, no. 2, 2006.

[10] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," IEEE Trans. on Knowl. and Data Eng., vol. 20, no. 8, 2008.

[11] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems, Berkeley, CA, USA, 2007, pp. 1-6.

[12] M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008.

[13] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," Cryptology ePrint Archive, Report 2009/081, 2009, http://eprint.iacr.org/.

[14] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in ESORICSâĂŹ09: Proceedings of the 14th European Conference on Research in Computer Security, Berlin, Heidelberg, 2009, pp. 355-370.

[15] K. Zeng, "Publicly verifiable remote data integrity," in Proceedings of the 10th International Conference on Information and Communications Security, ser. ICICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 419-434.

[16] A. F. Barsoum and M. A. Hasan, "On verifying dynamic multiple data copies over cloud servers," Cryptology ePrint Archive, Report 2011/447, 2011, 2011, http://eprint.iacr.org/

[17] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in ASIACRYPT '01:Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, London, UK, 2001, pp. 514-532.