Asian Journal of Information Technology 15 (6): 1056-1072, 2016
ISSN: 1682-3915

Integrity Verification for Dynamic Multi-Replica Data in Cloud Storage

Alshaimaa Abo-Alian, Nagwa L. Badr and M.F. Tolba
Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

Abstract: Despite of the cost effectiveness and high reliability of the cloud storage service, it has several security and privacy issues. As important as the service; it is vulnerable to attacks or outages which would bring irretrievable losses to data owners. It is a crucial requirement for the cloud service provider to provide data security practices to convince data owners that their outsourced data is available, correct and safe. Recently, many auditing schemes have been proposed to periodically verify the integrity of outsourced data in the cloud storage. However, most of these schemes audit static and single copy data files; they do not consider data dynamic operations on replicated data such as insertion, deletion and modification. Moreover, they lack the data recovery feature to repair corrupted data. This study, proposes a Multi Replica Dynamic Provable Data Possession (MR-DPDP) scheme, a public and privacy-preserving auditing scheme for replicated cloud storage which can also support efficient data recovery using Raptor codes. In addition, the proposed MR-DPDP scheme efficiently differentiates replicas by utilizing the Paillier probabilistic encryption scheme and consequently prevents cloud service providers from storing fewer replicas than have been paid for. The proposed scheme also supports efficient dynamic operations on data replicas over cloud servers. Performance analysis and experimental results prove the effectiveness and efficiency of the proposed scheme.

Key words: Auditing, cloud storage, cryptographic hash algorithms, fountain codes and homomorphic verifiable tags, MR-DPDP

INTRODUCTION

In recent years, cloud computing has been gaining growing interest from both business and academic organizations. Cloud computing, as defined by NIST is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or cloud provider interaction (Mell and Grance, 2011). The key advantages of using Cloud Computing are: Multi-tenancy, massive scalability, elasticity, pay as you go pricing model and self-provisioning of resources (Mather *et al*., 2009).

Cloud Storage Service (CSS) is an important service of cloud computing which allows data owners to move data from their local computing systems to remote Cloud Service Providers (CSPs) (Yang and Jia, 2012). CSS relieves the burden of storage management and maintenance. Moreover, it is cost effective as it makes data owners avoid the initial investment of an expensive infrastructure setup, large equipment and frequent maintenance costs (Abo-Alian *et al*., 2016b). Unfortunately, Cloud storage introduces new security challenges which make data owners worry about their data (Yang and Jia, 2012). Examples of these different security issues are:

- Data could be misused or accessed by unauthorized users,
- Cloud outages and security breaches of noteworthy cloud services appear from time to time such as; Amazon S3's recent downtime (Amazon.com, 2008) and Gmail's mass email deletion incident
- The cloud service providers may be dishonest and they may discard the data which has not been accessed or rarely accessed or maintain fewer replicas than have been paid for to save storage space
- The cloud service providers may choose to hide data loss and claim that data is still correctly stored in the cloud (Abo-Alian *et al*., 2016)

Since, the outsourced data is not always trustworthy due to the loss of physical control and possession over the data, it is crucial to have remote data auditing services that verify the integrity and availability of outsourced data on the cloud (Sookhak *et al*., 2015). The integrity of data storage can be effectively verified in traditional storage systems through the deployments of checksums, trap door hash functions, Message Authentication Codes (MAC), digital signatures, etc. (Liu *et al*., 2015). However, these methods cannot be directly applied to verify the

Corresponding Author: Alshaimaa Abo-Alian, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

integrity of cloud storage because they require a local copy of the outsourced data. Furthermore, it is impossible to download all the data in order to verify the data integrity due to the expensive I/O and transmission costs. Therefore, remote data auditing is a very resource demanding operation in terms of computational resources, memory space and communication costs.

Hence, many researchers have focused on efficiently achieving the integrity and reliable access of outsourced data by proposing various schemes to audit the data stored on CSS (Abo-Alian *et al*., 2015). The existing data storage auditing schemes can be classified into two main categories: Provable Data Possession (PDP) methods and Proof of Retrievability (PoR) methods. In PDP methods, in order to verify the integrity of data without sending it to untrusted servers, the auditor verifies probabilistic proofs of possession by sampling random sets of blocks from the cloud service provider (Yan and Jia, 2012). In PoR methods, a set of randomly-valued check blocks called sentinels are embedded into the encrypted file. For auditing the data storage, the auditor challenges the server by specifying the positions of a subset of sentinels and asking the server to return the associated sentinel values (Liu *et al*., 2011).

Several variations of PDP schemes (Anwar *et al*., 2012; Attas and Batrafi, 2011; Tripathi and Jalil, 2013; Kumar and Rajkumar, 2014; Thakur and Gupta, 2014) were proposed under different cryptographic assumptions. However, most of these schemes (Wang *et al*., 2013; Zhang and Blanton, 2013) deal with integrity checks and do not support data recovery in case the data is lost or corrupted. Many schemes (Cao *et al*., 2012; Chen and Curtmola, 2013; Shacham and Waters, 2013; Yuan and Yu, 2013) deal only with archival static data files and do not consider dynamic operations such as insert, delete and update. Some schemes only support private auditing such as Chen and Curtmola (2012, 2013) and Mukundan *et al*. (2012).

The main contributions of this study are: Firstly, proposing a public and privacy preserving auditing scheme, namely Multiple Replica-Dynamic Provable Data Possession (MR-DPDP), for cloud storage that is applicable to single copy and multiple copy dynamic data files. Therefore, the proposed MR-DPDP scheme relieves the data owner from the online and computational burdens. Moreover, it keeps the data confidential /invisible to the TPA during the auditing process by exploiting the SHA-256 hash algorithm and the Homomorphic Verifiable Tags (HVTs). Secondly, the proposed scheme supports auditing for multi-replica data files while assuring that the CSP stores all the replicas which are agreed upon in the SLA, by utilizing the Paillier Probabilistic encryption scheme. Thirdly, the proposed scheme maintains efficient data dynamic operations and verifies the integrity of these dynamic operations, by using an authenticated skip list because the authenticated skip list has efficient search and update times of. 0 (log n) Fourthly, supporting efficient data recovery to repair the corrupted data in the case of single copy data files. Finally, presenting theoretical security proofs, performance analysis and experimental results to prove that the proposed MR-DPDP scheme reduces the TPA computational cost and achieves better reliability, availability and scalability by supporting replication and data recovery.

Literature review: The system model of any auditing scheme consists of three entities as mentioned (Liu *et al*., 2013). The first entity is the data owner that has large data files to be stored on the cloud and can be either individual consumers or organizations. The second entity is the Cloud Storage Server (CSS) which is managed by a Cloud Service Provider (CSP) and has significant storage space and computation resources to maintain clients' data. The Third Party Auditor or Verifier (TPA) is the third entity which has expertise and capabilities to check the integrity of the data stored on the CSS.

Considering the role of the verifier in the model, all auditing schemes fall into two categories; private auditing and pubic auditing (Zheng and Xu, 2012). In private auditing, only data owners can challenge the CSS to verify the correctness of their outsourced data (Yang and Jia, 2012), (Ren *et al*., 2015). Unfortunately, private auditing schemes have two limitations: they impose an online burden on the data owner to verify data integrity and the data owner must have huge computational capabilities for auditing. In public auditing or third party auditing, data owners are able to delegate the auditing task to an independent Third Party Auditor (TPA) without devotion of their computational resources (Yang and Jia, 2012). However, pubic auditing schemes should guarantee that the TPA keeps no private information about the verified data.

For verifying the integrity of outsourced data in the cloud storage, various auditing schemes have been proposed which can be categorized into below form.

Single-copy auditing schemes: They focus on a single copy of data outsourced at the CSP.

Multiple-copy auditing schemes: They audit data that may be replicated on multiple servers at the same CSP or at different CSPs.

Single-copy auditing schemes: Shacham and Waters (2013) proposed two fast PoR schemes based on an homomorphic authenticator that enables the storage server to reduce the complexity of the auditing by aggregating the authentication tags of individual file blocks. The first scheme is built from BLS signatures and allows public auditing. The second scheme is built on pseudorandom functions and allows only private auditing but its response messages are shorter than those of the first scheme, i.e., 80 bits only. Both schemes use the Reed-Solomon erasure encoding method (Plank, 1997) to support an extra feature which allows the client to recover the data outsourced in the cloud. However, there are some drawbacks: Firstly, encoding and decoding are slow for large files. Secondly, the communication complexity for proof response is still linear to the number of elements in each erasure coded file. Thirdly, those schemes focus on static data files and do not support data dynamics.

Yuan and Yu (2013) managed to suppress the need for the tradeoff between communication costs and storage costs. They proposed a PoR scheme with public auditing and constant communication costs by combining techniques such as constant size polynomial commitment, BLS signatures and homomorphic linear authenticators. On the other hand, their data preparation process requires (s+3) exponentiation operations where s is the block size. However, their scheme does not support data dynamics.

Xu and Chang (2011) proposed a new PDP model called POS. POS requires no modular exponentiation in the setup phase and uses a smaller number, i.e., about 102 for 1G file, of group exponentiation in the verification phase. But, POS only supports private auditing for static data and its communication cost is linear in relation to the number of encoded elements in the challenge query.

Ateniese *et al*. (2011) introduced a PDP model which has two advantages: Lightweight and robustness. Their challenge/response protocol transmits a small, constant amount of data and this minimizes network communication. Furthermore, it incorporates mechanisms for mitigating arbitrary amounts of data corruption. On the other hand, their PDP model has three limitations: First, It supports only private auditing. Second, it does not support data dynamics. Finally, it relies on the Reed-Solomon encoding scheme in which the time required for encoding and decoding n-block files is $O(n^2)$.

Wang *et al*. (2011) proposed a public auditing protocol that supports fully dynamic data operations by manipulating the classic merkle hash tree construction for block tag authentication in order to achieve efficient data dynamics. They could also achieve batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the Third Party Auditor (TPA). Unfortunately, their protocol does not maintain the privacy of the data and TPA could derive users' data from the information collected during the auditing process. Additionally, their protocol does not support data recovery in the case of data corruption.

(Liu *et al*. (2013) addressed the security problem of the previous auditing protocol (Wang *et al*., 2011) in the signature generation phase which allows the CSP to cheat by using blocks from different files during verification. They presented a secure public auditing protocol based on the homomorphic hash function and BLS short signature scheme which is publicly verifiable, supports data dynamics and preserves privacy. However, their protocol suffers from massive computational and communication costs.

Liu *et al*. (2011) used a skip list structure to support data dynamics in their PDP model that reduces the computational and communication complexity from log (n) to constant. However, the use of a skip-list creates some additional storage overheads, i.e., about 3.7% of the original file at the CSP-side and 0.05% at the client-side. Additionally, it only supports private auditing.

Chen and Curtmola (2012) proposed a robust PDP that supports data dynamics and mitigates arbitrary amounts of data corruption. This PDP scheme reduces the high costs of the reed-solomon codes by using cauchy matrices and replaces insert/delete operations with append/modify operations when updating the RS-encoded parity data. However, their scheme supports only private auditing and causes high communication costs for updates.

Cao *et al*. (2012) proposed a public auditing scheme that is suitable for distributed storage systems with concurrent user access. They utilized the exact repair method so that no verification tags need to be generated on the fly for the repaired data. Consequently, it relieves the data owner from the online burden. However, their scheme introduces storage overheads at each server and uses an additional repair server to store the original packets besides the encoded packets. It also does not support data dynamics.

Zhang and Blanton (2013) proposed a new data structure called the "balanced update tree," to support dynamic operations while verifying data integrity. The size of this tree is independent of the overall file size as it

depends on the number of updates. However, it introduces increased storage overheads on the client and the auditing scheme requires the retrieval, i.e., downloading, of data blocks which leads to high communication costs.

Wang *et al.* (2013) proposed a privacy-preserving public auditing scheme using random masking and Homomorphic Linear Authenticators (HLAs) (Ateniese *et al.*, 2009). Their auditing scheme also supports data dynamics using the Merkle Hash Tree (MHT) and it enables the auditor to perform audits for multiple users simultaneously and efficiently. Unfortunately, their scheme is vulnerable to a TPA offline guessing attack.

Li *et al.* (2012, 2013) proposed a full data dynamic PDP scheme that uses an SN (Serial Number)-BN (Block Number) table to support data block updates. Then, they discussed how to extend their scheme to support other features, including public auditing, privacy preservation, fairness and multiple-replica checking.

Recently, Yang and Jia (2014) presented the Third-Party Storage Auditing Scheme (TSAS) which is a privacy preserving auditing protocol. It also supports data dynamic operations using an Index Table (ITable). They applied batch auditing for integrity verification for multiple data owners in order to reduce the computational cost on the auditor. However, this scheme moved the computational loads of the auditing from the TPA to the CSP, i.e., pairing operation which introduced high computational costs on the CSP side.

Multiple-copy auditing schemes: To improve system availability and reliability, the data owners may choose to replicate their critical data on multiple servers, so users can access the data from a nearby site (Sun *et al.*, 2012). Barsoum and Hasan (2011) (2012) proposed two dynamic multi-copy provable data possession schemes: Tree-based and map-based dynamic multi-copy provable data possession (TB-DMCPDP and MB-DMCPDP, respectively). These schemes prevent the CSP from cheating and maintaining fewer copies, using the diffusion property of an AES encryption scheme. The TB-DMCPDP scheme is based on the Merkle Hash Tree (MHT) whereas the MB-DMCPDP scheme is based on a map-version table to support outsourcing of the dynamic data. The setup cost of the TB-DMCPDP scheme is higher than that of the MB-DMCPDP scheme. On the other hand, the storage overhead is independent of the number of copies for the MB-DMCPDP scheme, while it is linear with the number of copies for the TB-DMCPDP scheme. The apparent limitations of both schemes are: High computational and communication costs. The authorized users should know the replica number in order to generate the original file.

Zhu *et al.* (2012) proposed a co-operative provable data possession scheme (CPDP) for multi-cloud storage integrity verification along with two fundamental techniques: Hash Index Hierarchy (HIH) and Homomorphic Verifiable Response (HVR). Under hash index hierarchy, multiple responses of the clients' challenges, computed from multiple CSPs, can be combined into a single response as the final result. Homomorphic verifiable response supports distributed cloud storage in a multi-cloud storage environment and implements an efficient collision-resistant hash function. However, Wang and Zhang (2014) proved that the CPDP is insecure because it does not satisfy the knowledge soundness, i.e., any malicious CSP or malicious organizer can generate the valid response which can pass the verification even if they have deleted all the stored data. Additionally, the CPDP does not support data dynamics.

Etemad and Kupcu (2013) proposed a distributed and replicated DPDP (DR-DPDP) that provides transparent distribution and replication of user data over multiple servers where the CSP may hide its internal structure from the client. This scheme uses persistent rank-based authenticated skip lists to handle data dynamic, i.e., insertion, deletion and modification, more efficiently. Their scheme supports dynamic version control which enables accessing older values of updated data. On the other hand, DR-DPDP has three noteworthy disadvantages: First, it only supports private auditing. Second, it does not support the recovery of corrupted data. Third, the organizer looks like a central entity that may get overloaded and can cause a bottleneck.

Mukundan *et al.* (2012) proposed a Dynamic Multi-Replica Provable Data Possession scheme (DMR-PDP) that uses the Paillier probabilistic encryption for replica differentiation so, that it prevents the CSP from cheating and maintaining fewer copies than have been paid for. The DMR-PDP also supports efficient dynamic operations such as block modification, insertion and deletion on data replicas over cloud servers. However, it supports only private auditing and does not provide any security proofs.

Chen and Curtmola (2013) proposed a remote data checking scheme for replication based distributed storage systems, called RDC-SR. It enables server side repair and places a minimal load on the data owner who only has to act as a repair coordinator. In RDC-SR, each replica constitutes a masked/encrypted version of the original file in order to support replica differentiation. In order to overcome the Replicate On The Fly (ROTF) attack, they make replica creation a more time consuming process. However, RDC-SR has three remarkable limitations: First, the authorized users must know the random numbers used

in the masking step in order to generate the original file. Second, it only supports private auditing. Third, it works only on static data.

MATERIALS AND METHODS

The Proposed Multi-Replica Dynamic Provable Data Possession (MR-DPDP) scheme: Most existing work on data auditing in cloud computing was designed based on a certain data redundancy feature; either a replicated data file or a single copy data file. There are some challenges and difficulties that must be overcome when auditing replicated data files:

- Efficient replica differentiation: Ensuring that the CSP is storing all data replicas agreed upon in order to prevent the CSP from storing fewer replicas than have been paid for/agreed on
- Efficient verification of dynamic operations on all replicas. On the other hand, the main challenge in auditing single copy data files is supporting data recovery in the case of data corruption. Only few existing auditing schemes that have the data recovery feature and most of them rely on optimal erasure codes such as Reed-Solomon (Plank, 1997) codes produce n = m/r (r<1) fragments where any fragments are sufficient to recover the original data object. Unfortunately, optimal codes are costly (in terms of memory usage, CPU time or both) when m is large. Therefore, this study proposes the Multiple Replica Dynamic Provable Data Possession (MR-DPDP) scheme for cloud storage to solve the aforementioned issues. This section first states some definitions applied in the design of the proposed auditing scheme. Then, it describes the architecture and the detailed processes of the proposed auditing scheme for cloud storage.

Notations and preliminaries: This sub-section lists some notations and provides details of the homomorphic verifiable tags, the paillier encryption scheme and raptor codes that are used in the proposed scheme. Assume that F is a data file to be outsourced and consists of a finite ordered set of m, blocks, i.e., $F = \{b_1 b_2,...b_m\}$. Let H: $\{0,1\} \rightarrow Z_n$be a full-domain hash algorithm and be a multiplicative cyclic group.

Homomorphic verifiable tags (HVTS): As stated in (Ateniese *et al*., 2011), given the message b (corresponding to a file block), T_b denotes its homomorphic verifiable tag. The file F that is represented as a set of blocks and its corresponding tags will be stored on the CSP. Homomorphic verifiable tags act as auditing metadata for the file blocks which have the following properties:

- Blockless verification; using HVTs, the CSP can generate a proof that allows the TPA to verify if the CSP correctly stores certain file blocks, even when the data owner does not have a local copy of the actual file blocks
- Homomorphic tags: given two values T_{bi} and T_{bj}, anyone can combine them into a value $T_{bi}+b_j$ corresponding to the sum of the messages b_i+b_j

Paillier encryption: Paillier (1999) cryptosystem is a homomorphic probabilistic encryption scheme which creates different ciphertexts each time the same message is encrypted using the same key. Using a public key (N, g), a ciphertext CT for plaintext PT is obtained as:

$$CT = g^{PT} x^{N} \bmod N \tag{1}$$

Using a secret key λ, plaintext can be obtained as:

$$PT = l\left(CT^{\lambda} \bmod N^2\right).\left(L\left(g^{\lambda} \bmod N^2\right)\right)^{-1} \bmod N \tag{2}$$

Where :

N = P×q and, p, q are two prime numbers
λ = LCM (p-1, q-1)
g = A random number such that its order is a multiple of N;g ∈ z_n.
X = A random number and g ∈ z_n
L (u) = (u = 1)/N

The proposed MR-DPDP scheme utilizes the Paillier probabilistic encryption scheme to generate unique differentiable replicas because it is semantically secure and has efficient encryption complexity when compared to other probabilistic encryption schemes as illustrated in Table 1.

Raptor codes: With traditional erasure correcting codes (e.g., Reed-Solomon), decoding data or repairing after corruption generates many I/O operations and requires transferring a large amount of information over the network (Jiekak *et al*., 2013). Moreover, it is required to fix the corruption rate a priori. Alternatively, rateless erasure

Table 1: The comparison between different probabilistic encryption schemes

Probabilistic	(Blum and Goldwasser, 1985)	(Paillier, 1999)	(Damgard *et al.*, 2010)
Encryption complexity	One modular multiplication	Two modular exponentiation and one multiplication	Three modular exponentiation and one multiplication
Decryption complexity	Four modular exponentiation and two modular multiplication	Two modular exponentiation	Two modular exponentiation
Ciphertext expansion	Longer than the plaintext by a constant number of bits (λ+1)	Twice the size of the plaintext	(1+1/s)times the size of the plaintext;while using computations modulo n^{s+1},n is an RSA modulus and s is a natural number.
Security	Semantically Secure; Assuming the intractability of integer factorization - Vulnerable to chosen ciphertext attack.	Semantically secure under the decisional composite residuosity assumption. No adaptive chosen ciphertext attacks.	The same security level as Paillier

- Raptor codes

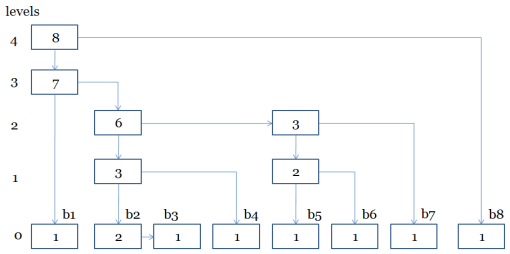

Fig. 1: An example of the Rank-based Authenticated Skip List (RASL)

codes such as raptor codes transform a data object of fragments into a practically infinite encoded form (Abu-Libdeh *et al.*, 2010). Therefore, Raptor codes are utilized for encoding single copy data files in the proposed scheme in order to support efficient data recovery. A raptor code (Shokrollahi, 2006) is a fountain code that encodes a message of symbols into a limitless sequence of encoding symbols, such that knowledge of any or more encoding symbols allows the message to be recovered with some non-zero probability. It is formed by the concatenation of two codes:

- Pre-code or outer code: A fixed-rate erasure code with a high rate to produce intermediate redundant blocks
- Inner code: Takes the intermediate blocks and generates a sequence of encoding symbols. Each encoding symbol is the XOR of a randomly chosen set of symbols from the intermediate blocks

A Raptor Code (Ho, 2003) is specified by parameters (k, c, Ω (x) where c is the (n,k) erasure correcting block code, called the pre-code and Ω (x) is the generator polynomial of the degree distribution of the LT code that is calculated as shown in Eq.3

$$\Omega(x) = \sum_{i=1}^{k} \Omega_i x^i \quad (3)$$

Where Ω_i is the probability that the degree of an output node is i.

Rank-Based authenticated skip lists: As defined in (Erway *et al.*, 2009), a Rank-based Authenticated Skip List (RASL) is a skip list in which every node v above the bottom level not only stores two pointers, namely right (v) and down (v) but also stores a label f (v) and a rank r (v). The label f (v) is computed using some collision-resistant cyptographic hash function h as a function of f (right (v))and f (dwn (v)) as follows:

Case 0: v = null

f (v) = 0;

Case 1: l (v) >0

f (v) = h (l (v),r (v), f (down (v))), f (right (v)) ;

Case 2: l (v) = 0

f (v) = h (l (v),r (v), x (v), f (right (v))). Where l (v) is the height of node v in the skip list (l (v)) = 0 for the nodes at the bottom level). The rank r (v) stores the number of nodes at the bottom level that can be reached from v.

The proposed scheme utilizes the RASL to support efficient authenticated dynamic operations on files at the block level such as insert, update and delete. An example of MRASL, constructed based on a file with 8 blocks is illustrated in Fig. 1 where each node stores a hash value calculated according to its level, rank, the hash value of the down neighbor and the hash value of the right neighbor. Unlike the Merkle Hash Tree (MHT), the RASL does not require continuous rebalancing operations that may degrade the performance of the dynamic operations. Moreover, in the worst case of insertion or deletion operations, the time taken to modify a block and the corresponding file tags is 0 (log n); where n is the number of file blocks because the RASL will only modify node

Table 2: Comparison of different cryptographic hash functions

Parameters	MD5	SHA-1	SHA-256	SHA-512
Message digest size (bits)	128	160	256	512
Message size (bits)	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$
Word size (bits)	32	32	32	64
Block size (bits)	512	512	512	1024
Number of steps	64	80	64	80
Complexity of the best attack	2^{64}	2^{80}	2^{128}	2^{256}

values on the path up to the root node. On the contrary, the index table may suffer a performance penalty during insertion or deletion operations because the indices of all the blocks after the insertion/deletion point are changed and all the tags of these blocks should be re-calculated which requires 0 (n) computations.

Cryptographic hash functions: Acryptographic hash functionis ahash function (i.e., that takes an arbitrary amount of data as input and produces an output of fixed size) that has the following properties (Menezes *et al*., 1996):

- Ease of computation; Given h and an input, x, h (x) is easy to compute
- Pre-image resistance: It is computationally infeasible to find any input from its hash value, i.e., given y, it is infeasible to find a pre-image x' such that h (x') = y
- 2nd pre-image resistance; It is computationally infeasible to find any sec input which has the same output as any specified input, i.e., given x, it is infeasible to find a 2nd-preimage x =x' such that h (x) = h (x)
- Collision resistance; It is computationally infeasible to find any two distinct inputs x, x' which hash to the same output, i.e., such that h (x) = h (x').

Cryptographic hash functions have many data security applications such as data integrity, authentication, fingerprinting, etc. In thisstudy, the SHA-256 which is an unkeyed cryptographic hash function is utilized in building the Rank based Authenticated Skip List (RASL) due to its efficiency and security as shown in Table 2.

The proposed architecture: The proposed MR-DPDP scheme is an auditing scheme of data storage security in cloud environments that is applicable to single copy data files and replicated data files as well. The design goals of the proposed scheme can be summarized as follows:

- Supporting public auditing for outsourced data integrity verification In order to relieve the data owners from the online and computational burdens of the auditing process, the proposed auditing scheme allows the data owners to delegate the auditing process to a Third Party Auditor (TPA) to verify the correctness of the outsourced data on demand
- Privacy Preserving Assurance; Ensuring that the TPA keeps no private information about the verified data, i.e., preventing the leakage of the verified data during the auditing process
- Blockless verification; Allowing the TPA to verify the correctness of the outsourced data on demand without possessing or retrieving a local copy of the data
- Supporting data dynamics; Supporting dynamic data operations on outsourced data files such as insertion, deletion and modification while maintaining the same level of data correctness assurance and guaranteeing data freshness
- Robustness; Supporting data recovery in case that the data is lost or corrupted
- Availability and reliability; Supporting replication as it plays a fundamental role to improve the reliability against data failures. Moreover, supporting auditing for distinguishable multi-replica data files to make sure that the CSP is storing all data replicas agreed upon
- Stateless verification; Where the TPA is not needed to maintain a state between audits
- Efficiency
- Minimum computation complexity at the CSP and the TPA; Minimum communication complexity between the CSP and the TPA; Minimum storage overheads on the CSP and the TPA

To achieve the aforementioned goals, the proposed architecture consists of two main sub-systems as shown in Fig. 2; data management sub-system and auditing sub-system.

The data management subsystem is responsible for generating the scheme keys, preparing the data before being moved to the cloud storage and maintaining data dynamics. So, it comprises five processes:

Key generation (keygen): This process is run by the data owner in order to generate the system and data owner's keys which will be used in all scheme processes. It takes as input security parameter and outputs; private key and public key for block tag generation and pseudorandom function key for replica generation, as illustrated

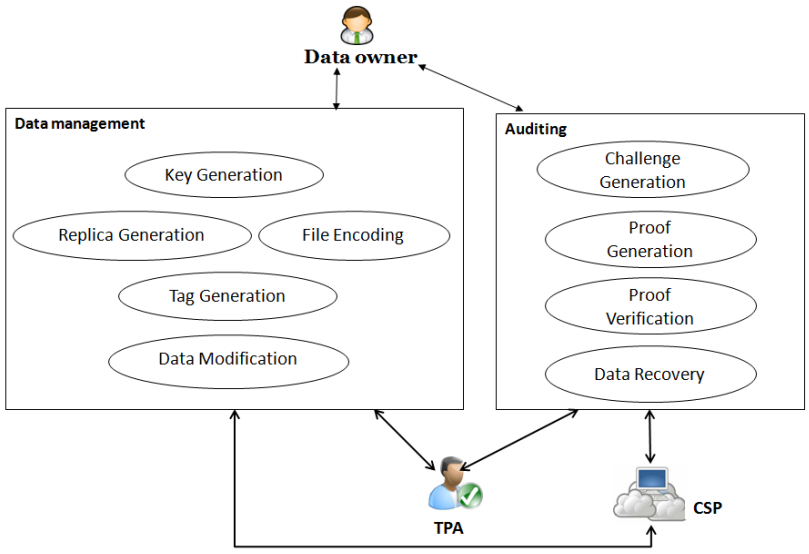

Fig. 2:Architecture of the proposed MR-DPDP scheme

The key generation process:
(sk, pk, sk_h, key $_{PRF}$) ← Key gen (1^{λ})

- Choose two random primes p and q in G
- Let N = pq be the RSA modulus φ (N) = (p-1) (q-1)
- Generate a random prime number e
- Set d = a^{-1} mod φ (N)
- Set private key sk = d
- Set public key pk = (N, e)
- Generate a Pseudorandom key key $_{PRF}$

Replica generation (Replica Gen): To improve system availability and reliability, the data owners may choose to replicate their critical data on multiple servers. This process generates unique differentiable replicas of the data file in order to prevent the CSP from storing fewer replicas than have been paid for. Unique copies of each file block is created by encrypting it using a homomorphic probabilistic encryption scheme which creates different ciphertexts each time the same data block is encrypted using the same key. This process is run by the data owner if the data owner chooses the multiple replica version. It takes the number of replicas and the file as input and generates unique differentiable replicas $\{F_r\}$ $1 \le r \le R$ as discussed. This algorithm is run only once. Unique copies of each file block of file are created by encrypting it using the Paillier probabilistic encryption scheme.

The replica generation process: $\{F_r\}$ $1 \le r \le R$ ← Replicagen (R, F). For each replica index r, where r = {1, R}: o Let $\{k_r\}$ represent the numbers generated using the key_{PRF} and x_{rj} represent any random number used in the Paillier encryption scheme. Compute $\{(1+b_1 N) (k_r x_{r1})^N, (1+b_2 N) (k_r x_{r2})^N, \ldots (1+b_m N) (k_r x_{rm})^N\}$.

File encoding (encode): When a data owner outsources a single copy data file, the file is encoded using any erasure correction codes before being outsourced. Encoded files can mitigate arbitrary amounts of data corruption. On the contrary, there is no need to encode the file before outsourcing in the case of multiple-replica data files; hence, a server's replica can be reconstructed even from a complete loss using further replicas on other servers. Therefore, this process is run by the data owner in order to encode the data file when the data owner chooses to outsource a single copy data file without replication. File encoding supports data recovery by adding more encoding symbols to the file that allows the file to be recovered in the case of any corruption. Raptor codes are utilized for encoding single copy data files. Accordingly, this process takes as input key $_{PRF}$, outputs the encoded file F' and the coding vector $\Delta_j$and works as illustrated

The file encoding process: (F', Δ_j) ← Encode (key $_{PRF}$, F) Let F = $\{b_1, b_2, \ldots, b_m\}$ be the file to be encoded, where each b_j is a data block of s bits. Encode F by an erasure code (pre-code) to obtain F' = $\{y_1, \ldots y_m\}$. Choose a random s×s binary matrix A = $[A_1 \; A_s]^T$, where each A_j is an s bit vector. For each $1 \le i \le k$, Create authenticators $\delta_1, \ldots, \delta_k$ as $\delta_i = PRF_{Key\ PRF}(I) \oplus Y_I$. Output the encoded file F' = $\{y_1, \ldots, y_m, \delta_1, \ldots, \delta_n\}$ output a coding vector Δ_j is attached where each bit represents whether the corresponding original block is combined into F' or not.

Tag generation (taggen): Before outsourcing the generated replicas or the encoded file to the CSP, the tag generation process pre-computes a set of integrity tags

for each data block for further data integrity auditing. These tags will be used by the TPA to verify the integrity of the outsourced data by requesting a number of randomly selected blocks and their corresponding tags to be returned, as described later in the challenge process. The Tag Gen process is run by the data owner. It takes the private key and the unique differentiable file replicas $\{F_{r\ 1\le r\le R}\}$or the encoded file F' as inputs. It outputs the tags set $\Phi = {}_{1\le j\le m}$ which is an ordered collection of tags for the data blocks.

In this process, the data owner also constructs a Rank based Authenticated Skip List (RASL) for the encoded file or all replicas where the bottom-level node of the skip list is the tag of the block . At the end of this process, the data owner sends the set of tags and the start node of the RASL to the TPA and sends the data blocks along with their tags to the CSP. Then, the data owner may delete the data files and the corresponding tags from the local storage. The study illustrates a detailed description of the TagGen process. It is valuable to note that the file identifier is embedded into the block tag to prevent the CSP from cheating and using data blocks from different files and passing the audit. Since, the proposed scheme supports public auditing, it should be privacy-preserving (i.e., TPA should not be able to derive users' data during auditing). Thus, the private key is used in the tag generation process rather than the public key (N, e) because the private key is only known by the data owner. As a result, the TPA cannot derive the data blocks during the audit process which in turn, assures the privacy preserving feature of the proposed auditing scheme.

The tag generation process: TagGen for Single-Copy file: Divide File F into an ordered collection of blocks $\{b_j\}$; $1 \le j \le m$. Generate a tag for each block b_j as follows: $\sigma_j \leftarrow (H(F_{ID}\backslash\backslash j).u^{b_j})^{sk} \bmod N$, $\Phi = \{\sigma_j\}_{1\le j\le m}$ Where F_{ID} is the file identifier; F_{ID} ,is a random number. Construct a modified rank-based authenticated list RASL for F' . Send the tags set Φ and the start node SN of the RBAL to the TPA. Send the data blocks $\{b_j\}$ along with their tags Φ to the CSP and delete them from the local storage.

Taggen for multiple-replica file: $\Phi \leftarrow$ Tag Gen for multiple $(sk, \{F_r\}_{1\le r\le R})$. Divide each distinct file replica F_r into an ordered collection of blocks $\{b_{rj}\}_{;1\le j\le m}$. Generate a tag for each blocks b_{rj} I as follows: $\sigma_{rj} \leftarrow (H(F_{ID}//j//i)).u^{b_{rj}})^{sk} \bmod N$ where F_{ID} is the file identifier ; F_{ID} = filename $\|m\|u$, u is a random number. Generate an aggregate tag σ_j for the blocks at the same indices In all replicas as $\sigma j = \prod_{r=1}^{R} \sigma_{rj}$ Construct a modified rank -based authenticated list RASL for all file replicas. Send the tags set $\Phi = \{\sigma_j\}_{1\le j\le m}$ and the start node SN of the RASL to the TPA send the data blocks $\{b_{rj}\}$ along with their tags Φ to the CSP and delete them from the local storage.

Data modification (modify): This process is responsible for performing the modification requests and checking whether the CSP has performed the modification on all the replicas' blocks and their corresponding tags as well. In this process, the RASL is used to efficiently support dynamic operations on replicated data files and encoded single-copy data files as well. The data owner sends a modification request that contains the type of modification, blocks to be modified and the new block values. Then, the CSP modifies the underlying blocks, re-compute the corresponding tags and modifies the corresponding nodes in the RASL. Finally, the TPA challenges the CSP with the modified block indices in order to verify whether the CSP has performed the modification correctly or not. More formally, this process takes as input; the index of the block to be modified j, the new block value b_j and the type of the dynamic operation Op; i.e., 0 for deletion, 1 for insertion and 2 for update. If a user wants to insert a new block b_j after the jth block, a new tag σ_j will be computed and inserted in the RASL after the jth node. Also, node values on the path up to the start node are re-computed. If a user wants to delete block, the corresponding node and tag σj will be deleted from the RASL. Also, node values on the path up to the start node are re-computed. If a user wants to update the value of the jth block to be b_j A new tag σj will be computed. Then, the old tag σj is replaced with the new tag σj and the node values on the path up to the start node are re-computed. After performing the modification request, the TPA challenges the CSP with the modified block indices in order to verify whether the CSP has performed the modification correctly or not.

The auditing sub-system is concerned with frequently verifying the integrity of the outsourced data without downloading the entire data files or revealing any private information about the verified data. It supports efficient auditing of dynamic data operations for multiple replicas or single copy data files. The auditing sub-system is also responsible for notifying the data owner if there is any corruption and data recovery in order to repair the corrupted data. This sub-system is composed of four processes:

Challenge generation (challenge): In this process, the TPA challenges the CSP to verify the integrity of all outsourced replicas. This process randomly specifies the positions of the blocks to be checked. The number of blocks to be challenged is determined by the preferred detection probability of file corruption. In other words, achieving high detection probability of file corruption requires challenging a considerable number of data blocks as discussed in Section 5.1.1. Thus, the TPA sends

(number of blocks to be challenged; $1 \le c \le m$) and two distinct PRF keys at each challenge: k_1 and k_2. The Pseudo-Random Function (PRF) keyed with k_1 is use$_d$ to generate c random indices which indicate to the file blocks that the CSP should use to prove the integrity. The PRF keyed with k_2 is use$_d$ to generate y_j random values that are associated with each random index j and are used by the CSP while generating the response. Then, the challenge set $Q = \{(j,y_j)\}$ of the pairs of random indices and values is generated at the CSP.

Proof generation (prove): This process is run by the CSP, upon receiving the challenge set Q, to generate a proof that is used in the verification equation. In this process, the CSP proves that it is still correctly storing all file blocks by sending a tag proof and a data proof as well without revealing any information about the challenged data blocks to the TPA. In the case of a single copy file, the CSP computes the data proof μ and the tag proof for the challenged blocks as follows:

$$\mu = \sum_{(j,y_j)\in Q} y_j b_j \tag{4}$$

$$\sigma = \left(\prod_{(j,y_j)\in Q} \sigma_j \right)^{y_j} \bmod N \tag{5}$$

Where in the multiple-copy version, the CSP computes a data proof for each challenged replica block as follows:

$$\mu_r = \sum_{(j,y_j)\in Q} y_j b_{rj} 1 \le r \le R \tag{6}$$

Then, the CSP aggregates all the data proofs of all challenged blocks into a single value as follows:

$$\mu = \prod_{r=1}^{R} \mu_r, \tag{7}$$

computes the tag proof using Eq. 5 and finally sends the proof to the TPA.

Proof verification (verify): Upon receiving the proofs from the CSP, the TPA runs this process in order to check whether the proofs are valid or not. If there is any corruption, the verification equations will not hold and the TPA should notify the data owner and run the data recovery process. Therefore, this process takes as the challenge set Q, the file identifier F_{ID} and the proof P returned from the CSP. By parsing the F_{ID}, the TPAcan recover the filename and the random number u. The TPA checks the following verification equation in the case of single copy data file:

$$\sigma^e \Leftrightarrow \prod_{(j,y_j)\in Q} H(F_{ID} \| j)^{y_j} u^{\mu} \bmod N \tag{8}$$

and this equation:

$$\sigma^e \Leftrightarrow \left(\prod_{(j,y_j)\in Q} H(F_{ID} \| j \| i)^{y_j} \right)^{R} u^{\mu} \bmod N \tag{9}$$

in the case of replicated data file and outputs TRUE if the verification equation passed or FALSE otherwise.

Data recovery (recover): This process is executed when the TPA has detected a corrupted replica or file during the proof verification process. In the case of multiple-replica data files, the CSP could recover the original file from a healthy replica and restore the corrupted replica. In the case of single-copy encoded data files, the corrupted data can be recovered by the parity data in the raptor codes. To repair a corruption on the l-storage server, the TPA uses the corresponding coding vectors $\{\Delta_{lj}\}_{1 \le j \le n}$ to generate the encoded blocks $\{b_{lj}\}_{1 \le j \le n}$ and F_{lj} is generated by the XOR combination of $|\Delta_{lj}|$ original blocks as follows:

$$F_{lj} = b_{l1} \oplus \ldots \oplus b_{l|\Delta lj|} \tag{10}$$

RESULTS AND DISCUSSION

Security and performance analysis

Security analysis: This section briefly describes the security proof for our proposed. Recalling the security model defined in (Shacham and Waters, 2013). On the assumption that the TPA is honest-but-curious. It performs honestly during the whole auditing protocol but it is curious about the challenged data. On the other hand, the CSP is assumed to be dishonest and potentially malicious. The proposed auditing scheme should be correct and sound. For the correctness of the proposed scheme, the proof verification (verify) process should accept a valid proof generated from all key pairs (pk, sk) output by the KeyGen process, all files $F \in \{0,1\}$, all encoded files F with file identifier F_{ID} and authentication tags $\Phi = \{\sigma_j\} 1 \le j \le n$. The verify process accepts when interacting with the valid prover, i.e., the CSP returns a valid response:

$$\left(\text{verify}(pk, F_{ID}) \rightleftharpoons \text{Response}(pk, F_{ID}, Q, F) = 1\right) \tag{11}$$

For the soundness of our scheme, if any malicious CSP can generate a proof and convince the verify process that it is actually storing that file F correctly, it has to yield

up the right file F to an extractor algorithm for the proof generation. By using the security game of (Shacham and Waters, 2013) for the soundness of the proposed auditing scheme, in which the adversary is trying to cheat the verifier successfully by generating valid responses and passing the data verification without being detected.

Let ∇ = (Keygen,TagGen, Prove, Verify) be an auditing scheme and A be a probabilistic polynomial-time adversary. Consider the following security game among the system, a challenger and A.

- The system runsKeyGen (1^{λ}) → (pk,sk) and gives pk to the adversary A
- The adversary A chooses a data file F and sends it to the system. The system then runs TagGen (sk, F) →(σ, F)and responds the encoded data file F together with the authentication tag σ back to A
- With regard to the data file F chosen by adversary A, the challenger generates a random challenge message Q and sends it to A
- According to the received challenge Q, the adversary A generates a file F" F" = F', since, he might have modified or lost some part of F'. A then produces a proof response P by running an arbitrary algorithm art (F', σ, pk) → PRF rather than the Prove algorithm. The proof response Prf is sent back to the challenger
- The challenger verifies p by running the verify algorithm with p and pk. The output of verify (p,pk) is denoted as Rst
- The adversary A wins the game if and only if A can produce a prf with data file F', F' = F' and make the challenger generate Rst as TRUE through the verify algorithm.

∇ can be considered sound if any probabilistic polynomial-time adversary A can win the game with at most a negligible probability. The following security analysis depends on the intractability of the integer factorization problem (Menezes *et al*., 1996) and the RSA problem (Menezes *et al*., 1996).

Integer factorization problem: Given a positive integer n, find its prime factorization; that is write $n = p^{e1}{}_1\, p^{e2}{}_2 \ldots p^{ek}{}_k$ where the p_i are pairwise distinct primes and each $e_i \geq 1$.

RSA problem: Given a positive integer that is a product of two distinct odd primes p and q, a positive integer e such that gcd (e,(p-1) (q-1)) = 1and an integer c, find an integer m such that $m^e \equiv c \pmod n$.

Theorem 1 proves that the CSP can provide valid responses to the TPA's challenges only if it actually has all file replicas and they are intact.

Theorem 1: If the signature scheme used for file tags is existentially unforgeable and the RSA problem with large public exponents is hard, then in the random oracle model, the CSP cannot pass the verification of the proposed MR-DPDP scheme unless it responds with correctly computed proof for the challenge set.

Proof: First, since F_{ID} is embedded into the block tag, the CSP cannot use blocks from different file for proof generation and pass the verification algorithm even if the data owner uses the same secret key sk with all her files. That is due to the collision-resistance property of any cryptographic hash algorithm, i.e., it is impossible for a hash function to get two same hash values for different messages in the random oracle model.

Second, due to the different value of the random number u for each data block, the CSP cannot use the previous proof or old versions of challenged blocks to generate the new proof. Third, depending on the intractability of integer factorization problem, the TPA cannot get any information about the challenged blocks because it is computationally difficult to solve the following equation to get the data block b_{ij} where:

$$\frac{\prod_{(j,yj)\in Q} H\left(F_{ID} \,\|\, j \,\|\, T_j\right)^{yj}.U^{u} \bmod N}{\left(\prod (j,yj) \in Q(\sigma j)^{yj} \bmod N\right)^{e}} = 1 \qquad (12)$$

Where:

$$\mu = \prod_{r=1}^{R} \sum_{(j,y_j)\in Q} yjb_{ij} \qquad (13)$$

Finally, if any of the challenged data blocks or their tags are corrupted or not fresh on the CSP, the CSP cannot pass the auditing because the verification equation cannot hold.

Table 3 illustrates the different assumptions used in the security proofs of our proposed scheme versus the state of the art.

Table 3: Security proofs of the proposed MR-DPDP scheme and the state-of-the-art

Scheme proofs	Assumption of the security
Liu *et al.* (2013) and Wang *et al.* (2013)	Computational Diffie Hellman (CDH)
Yuan and Yu (013)	Computational Diffie Hellman (CDH), Strong Diffie Hellman (SDH) and Bilinear Strong Diffie Hellman (BSDH)
Yang and Jia (014)	Computational Bilinear Diffie Hellman (CBDH)
Etemad and Kupcu (2013)	The existence of a collision-resistant hash function and integer factorization
Barsoum and Hasan (2012)	Computational Diffie Hellman (CDH) and Discrete Logarithm (DL)
Proposed scheme	Integer factorization and RSA

Table 4: Performance analysis of the proposed MR-DPDP scheme and the state of the art

Scheme	Public auditing	Dynamic data	Replication	Data recovery	Computational costs CSP	TPA	Storage overheads CSP	TPA	Communication costs
Liu *et al.* (2013)	Yes	Yes	No	No	C_{SE}+O(c) (C_{mul}+C_{exp}+C_{add})	O(c) (C_{mul}+C_h +C_{exp})+2C_e	O((1+1/s)m)	O(1)	0(1)
Yuan and Yu (2013)	Yes	No	No	Yes	O(c+z) (C_{mul}+C_{exp})	O(c) (C_{mul}+C_{exp})+4$_e$	O((1+1/z)m)	O(1)	O(1)
Wang *et al.* (2013)	Yes	Yes	No	No	O(c) (C_{mul}+C_{exp} +C_{add})+C_h+C_e	O(c) (C_{mul}+C_h+C_{exp}) +2C_e	O((1+1/s)m)	O(1+1/s	O(sc)
Yang and Jia (2014)	Yes	Yes	No	No	O(c) (C_{add}+C_{mul}+C_{exp}) +O(s) (C_{exp}+C_s)	O(c) (C_h+C_{exp}+C_{mul}) +2C_e)	O(zm/s)	O(1)	O(c)
Etemad and Kupcu (2013)	No	Yes	Yes	No	O(1+logmR) (C_{exp}+C_{mul})	O(1+log mR) (C_{exp}+C_{mul})	O(log mR)	O(1)	O(log mR)
Barsoum and Hasan (2012)	Yes	Yes	Yes	No	O(c) (C_{add}+C_{mul}+C_{exp}) (C_{mul}+C_{exp})+O(R)C_{add}+2C_e	O(c) (C_h+C_{mul} +C_{exp})+O(s)	O(2m)	O(2m)	O(sR)
Proposed scheme (single copy)	Yes	Yes	No	Yes	O(c) (C_{add}+C_{mul}+C_{exp}) +C_{mod}	O(c) (C_{mul}+C_{exp} +C_h)+C_{mod}	O(2m)	O(m)	O(log m)
Proposed scheme (Multi-replica)	Yes	Yes	Yes	Yes	O(c) (C_{add}+C_{mul}+C_{exp}) +O(R)C_{mul}	O(c) (C_{mul}+C_{exp} +C_h)+C_{mod}	O(2m)	O(m)	O(log mR)

Performance analysis: This section evaluates the performance of the proposed MR-DPDP scheme. This section also compares the proposed MR-DPDP scheme with state-of-the-art. Table 4 illustrates the performance of the proposed auditing scheme in comparison with other auditing schemes. Suppose that there are R replicas of the outsourced data file F and F is divided into m blocks. Assuming that each block is of size z, let c be the number of challenged blocks. In Table 4, C_{exp} denotes the time cost of exponentiation in the group G; C_{mul} denotes the time cost of multiplication in the group G; C_e denotes the time cost of bilinear pairing; C_h denotes the time cost of the hash function H; C_{mod} denotes the time cost of modular operation in the group G; C_{add} denotes the time cost of addition in the group G and C_{SE} denotes the time cost of stream encryption. In other schemes, a data block may be divided into sectors.

As in Table 4, the proposed MR-DPDP scheme out performs other auditing schemes as it supports public auditing, data dynamic operations, replication and data recovery. Moreover, the proposed scheme achieves efficient verification cost (i.e., the compuational cost at the TPA) as it eliminates the usage of expensive pairing operations (i.e., C_e) unlike the other schemes (Liu *et al.*, 2013; Yuan and Yu, 2013; Yang and Jia, 2014; Barsoum and Hasan, 2012). In the multiple-copy version, the CSP computational cost of our scheme is comparable to that of (Barsoum and Hasan, 2012). The computational cost of our scheme on the TPA side is less than that of (Hasan, 2012) because of eliminating the usage of expensive pairing operations. Additionally, the verification cost at the TPA side does not depend on the number of replicas because of the aggregation of proof responses from the challenged servers while generating the proof.

On the other hand, the verification cost of the auditing schemes in (Barsoum and Hasan, 2012) grows linearly with the number of replicas. Although, the schemes (Liu *et al.*, 2013; Yuan and Yu, 2013) achieve constant communication costs and constant storage overheads at the TPA, they do not support auditing for replicated data files.

Implementation and experimental evaluation: The experiments are conducted using MATLAB and Java on a system with an Intel Core i5 processor running at 2.2 GHz and 4 GB RAM running Windows 7. In our implementation, we use Java Pairing Based Cryptography (JPBC) library version 2.0.0. To achieve an 80-bit security parameter, the elliptic curve group used has a 160-bit group order and the size of modulus N is 1024 bits. All files used in the experiments are downloaded from the Human Genome Project at NCBI (http://www. ncbi.nlm. nih.gov/genome/, 2014). All results are the averages of 20 trials.

Several experiments are conducted on the proposed scheme to explore the impact of the following parameters on the computation time at the TPA, the computation time at the CSP and the storage overheads:

- The corruption rate
- The challenge size
- The file size
- The block size
- The number of replicas

Moreover, some experiments are conducted to compare the proposed system with the state-of-the-art.

Parametric evaluation: In this sub-section, different parameters are examined to evaluate their impact on the efficiency of the MR-DPDP scheme.

The impact of corruption rate: Figure 3 shows the computational time (in seconds) at the CSP and the TPA

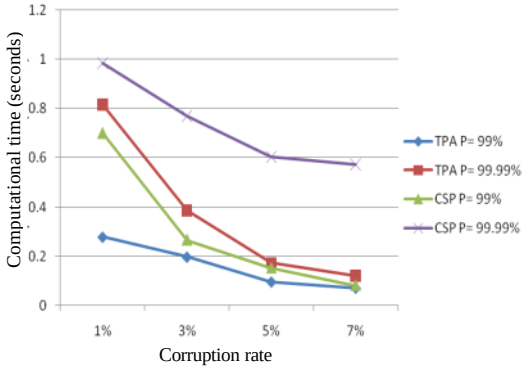

Fig. 3: Computational time of the proposed MR-DPDP scheme using different corruption rates and detection probabilities

Table 5: The storage overheads of the proposed MR-DPDP scheme using different file sizes and block sizes

File size	Storage overhead (KB)		
	z = 16 KB	z = 80 KB	z = 160KB
28 MB	1792	358.4	179.2
256 MB	16384	3276.8	1638
512 MB	32768	6553.6	3277
1 GB	65536	12800	6554

using a single-copy file, block size , different corruption rates p and different detection probabilities p. It is observed that the verification time (the computational time at the TPA) and the proof generation time (the computational time at the CSP) increase for high detection probabilities and decreases for high corruption rates because the auditing scheme has to challenge a high number of blocks in order to achieve a high detection probability. More formally, the challenge generation process computes the number of challenged blocks according to the corruption rate and the detection probability as follows: Given the corruption rate p and the number of challenged blocks c, the TPA can detect the CSP misbehavior with probability p . $P = 1-(1-p)^c$ Thus, the number of challenged blocks grows linearly with the detection probability and accordingly the verification time increases. Therefore, the computational time of the proposed scheme is proportional to the number of challenged blocks.

The impact of file size and block size: Table 5 illustrates how the file size and the block size affect the storage overheads at the TPA side while Fig. 4 illustrates the impact of the file size and the block size on the computational time at the TPA and the CSP sides, when using a corruption rate = 1% and a detection probability = 99.99%.

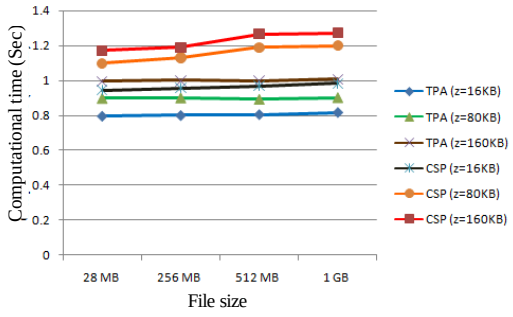

Fig. 4: The computational time of the proposed scheme using different file sizes and block sizes

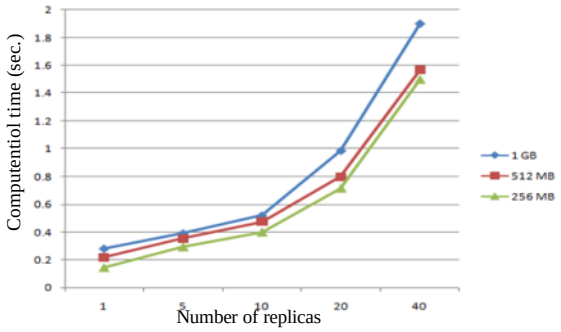

Fig.5: The setup computational time of the proposed MR-DPDP scheme using different number of replicas and file size

As shown in Fig. 4, the computational time at the TPA and the CSP sides are independent of the file size. However, the computational time grows linearly with the block size while the storage overheads decrease for higher block sizes.

The impact of number of replicas: Figure 5 illustrates the computational time of the proposed MR-DPDP scheme at the CSP, TPA and the data owner sides using different file sizes and different numbers of replicas with a block size = 16 KB, a corruption rate = 1% and a detection probability = 99.99%. As illustrated in Fig.5 the setup cost at the data owner side grows linearly with the number of replicas and the file size and the same for the computation cost at the CSP side, refer to Fig. 6. On the contrary, the computational time at the TPA side is independent of the number of replicas due to the fast verification time of RSA signature and the efficient decryption of the Paillier probabilistic scheme as shown in Fig. 7.

Comparative evaluation: This sub-section gives the computational time comparison between the MR-DPDP

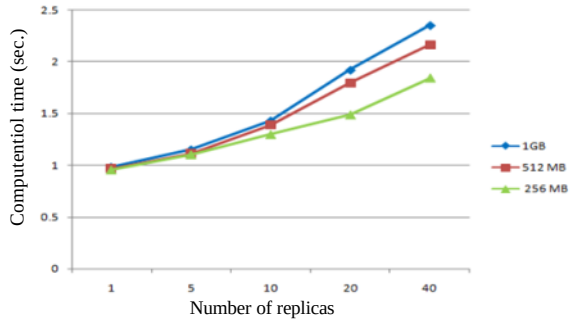

Fig. 6: The CSP computational time of the propsed MR-DPDP scheme using different of replicas and file sizes

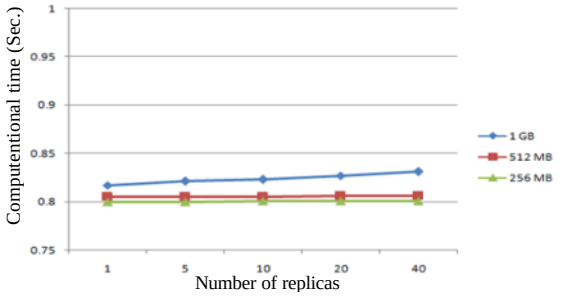

Fig. 7: The TPA computational time of our propsed scheme using different of replicas and file sizes

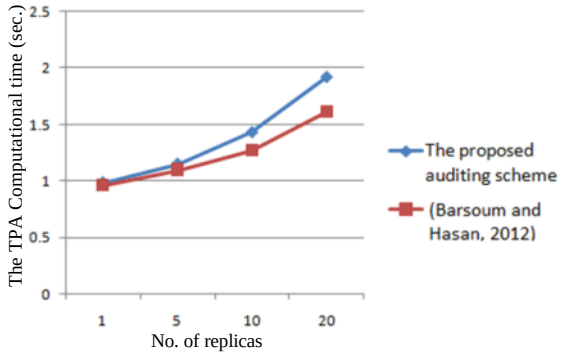

Fig. 8: The CSP computational time of our propsed auditing scheme (Barsoum and Hasan, 2012)

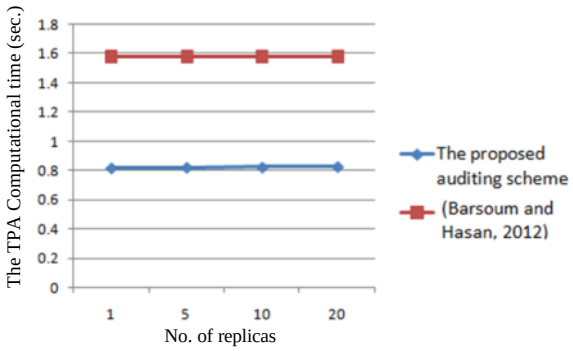

Fig. 9: The TPA computational time of our propsed auditing scheme (Barsoum and Hasan, 2012)

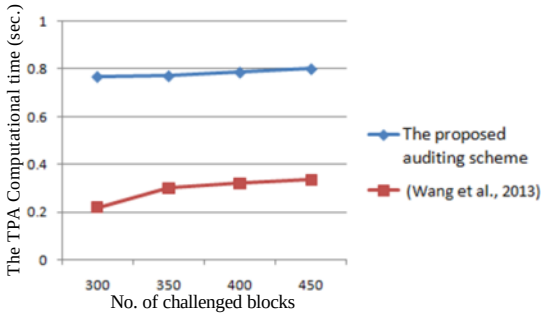

Fig. 10: The CSP computational time of the propsed auditing scheme (Wang *et al*., 2013)

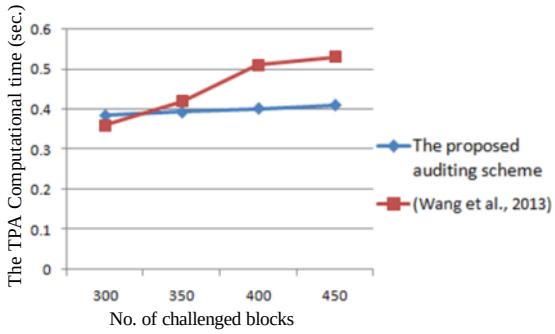

Fig. 11: The TPA computational time of the propsed auditing scheme and (Wang *et al*., 2013)

scheme and two existing schemes: The MB-DMCPDP proposed by Barsoum and Hasan (2012) and the auditing scheme proposed by Wang *et al*. (2013). Figure 8 and 9 compares the computational time of the proposed MR-DPDP scheme (Multiple replica version) with that of the scheme using different numbers of replicas, 1 GB file, 16 KB block size, corruption rate = 1% and detection probability = 99.99%. The CSP computational time of the (Barsoum and Hasan, 2012) scheme is less than that of the proposed scheme because the aggregation of the responses is moved from the challenged servers to the CSP in the proof generation process. That response aggregation reduces the verification computation cost of the proposed scheme at the TPA side. Therefore, the TPA computational time of the proposed scheme is less than that of (Barsoum and Hasan, 2012) because the TPA computational time of the proposed scheme is independent of the number of replicas and performs efficiently for larger number of replicas. Using a 1 GB file with a block size 16KB and 1% corruption rate, the computational time of the proposed scheme (single-copy version) is compared with that of the (Wang *et al*., 2013) scheme using different numbers of challenged blocks. As illustrated in Fig. 10 and11, the

proposed scheme reduces the TPA computational time because it eliminates the expensive computation of bilinear pairing while the verification in the (Wang *et al*., 2013) scheme uses two bilinear pairings.

CONCLUSION

In this study, a new privacy-preserving public auditing scheme is proposed for verifying the integrity of replicated data and single copy data in cloud storage. The proposed scheme utilizes the homomorphic verifiable tags and cryptographic hash algorithm to guarantee that the TPA would not learn any knowledge about the data content stored on the CSP during the auditing process. Moreover, the proposed scheme achieves efficient verification costs as the computational time at the auditor side is independent of the number of replicas. The proposed scheme supports replica differentiation to prevent the CSP from cheating and maintaining fewer replicas than have been paid for. Additionally, the proposed scheme supports efficient verification of dynamic data operations such as insertion, deletion and update on all underlying replicas. The proposed scheme benefits from the efficient encoding and decoding of Raptor codes to support data recovery for single copy data files. The experimental results, security and performance analysis show that the proposed scheme is complete, provably secure and efficiently comparable to existing schemes.

REFERENCES

Abo-Alian, A., N.L. Badr and M.F. Tolba, 2015. Auditing-as-a-service for cloud storage. Proceedings of the 7th IEEE International Conference Intelligent Systems, September 24-26, 2014, Warsaw, Poland, pp: 559-568.

Abo-Alian, A., N.L. Badr and M.F. Tolba, 2016a. Hierarchical attribute-role based access control for cloud computing. Proceedings of the 1st International Conference on Advanced Intelligent System and Informatics, November 28-30, 2015, Beni Suef, Egypt, pp: 381-389.

Abo-Alian, A., N.L. Badr and M.F. Tolba, 2016b. Keystroke dynamics-based user authentication service for cloud computing. Concurrency Computat.: Pract. Exper., 28: 2567-2585.

Abu-Libdeh, H., L. Princehouse and H. Weatherspoon, 2010. RACS: A case for cloud storage diversity. Proceedings of the 1st ACM Symposium on Cloud Computing, June 10-11, 2010, Indianapolis, IN., pp: 229-240.

Amazon.com, 2008. Amazon S3 availability event: July 20, 2008. http://status.aws.amazon.com/s3-20080720.html

Anwar, M.J., M.Y. Javed, S. Rehman and N. Asad, 2012. Applying provable data possession with elgamal in cloud computing. J. Basic Applied Sci. Res., 2: 7091-7094.

Ateniese, G., R. Burns, R. Curtmola, J. Herring and O. Khan *et al*., 2011. Remote data checking using provable data possession. ACM Trans. Inform. Syst. Security, Vol. 14. 10.1145/1952982.1952994

Ateniese, G., S. Kamara and J. Katz, 2009. Proofs of storage from homomorphic identification protocols. Proceedings of the 15th International Conference on the Theory and Application of Cryptology and Information Security, December 6-10, 2009, Tokyo, Japan, pp: 319-333.

Attas, D. and O. Batrafi, 2011. Efficient integrity checking technique for securing client data in cloud computing. Int. J. Electr. Comput. Sci., 11: 43-48.

Barsoum, A.F. and M.A. Hasan, 2011. On verifying dynamic multiple data copies over cloud servers. IACR Cryptol. ePrint Arch., Vol. 2011.

Barsoum, A.F. and M.A. Hasan, 2012. Integrity verification of multiple data copies over untrusted cloud servers. Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, May 13-16, 2012, Ottawa, Canada, pp: 829-834.

Blum, M. and S. Goldwasser, 1985. An Efficient Probabilistic Public-Key Encryption Scheme which Hides all Partial Information. In: Advances in Cryptology, Blakley, G.R. and D. Chaum (Eds.). Springer, Berlin, Heidelberg, ISBN: 978-3-540-15658-1, pp: 289-299.

Cao, N., S. Yu, Z. Yang, W. Lou and Y.T. Hou, 2012. LT codes-based secure and reliable cloud storage service. Proceedings of the IEEE INFOCOM, March 25-30, 2012, Orlando, FL., pp: 693-701.

Chen, B. and R. Curtmola, 2012. Robust dynamic provable data possession. Proceedings of the 32nd International IEEE Conference on Distributed Computing Systems Workshops, June 18-21, 2012, Macau, China, pp: 515-525.

Chen, B. and R. Curtmola, 2013. Towards self-repairing replication-based storage systems using untrusted clouds. Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy, February 18-20, 2013, San Antonio, TX., USA., pp: 377-388.

Damgard, I., M. Jurik and J.B. Nielsen, 2010. A generalization of paillier's public-key system with applications to electronic voting. Int. J. Inform. Security, 9: 371-385.

Erway, C., A. Kupcu, C. Papamanthou and R. Tamassia, 2009. Dynamic provable data possession. Proceedings of the 16th ACM Conference on Computer and Communications Security, November 9-13, 2009, Chicago, IL., USA., pp: 213-222.

Etemad, M. and A. Kupcu, 2013. Transparent, distributed and replicated dynamic provable data possession. Proceedings of the 11th International Conference on Applied Cryptography and Network Security, June 25-28, 2013, Banff, AB., Canada, pp: 1-18.

Ho, T., 2003. Summary of raptor codes. Scientific Commons.

Jiekak, S., A.M. Kermarrec, N. Le Scouarnec, G. Straub and A. van Kempen, 2013. Regenerating codes: A system perspective. ACM SIGOPS Operat. Syst. Rev., 47: 23-32.

Kumar, S.S. and M.N. Rajkumar, 2014. Secured dynamic auditing in cloud storage using fully homomarphic mechanism. Int. J. Software Hardware Res. Eng., 2: 207-211.

Li, C., Y. Chen, P. Tan and G. Yang, 2012. An efficient provable data possession scheme with data dynamics. Proceedings of the International Conference on Computer Science and Service System, August 11-13, 2012, Nanjing, China, pp: 706-710.

Li, C., Y. Chen, P. Tan and G. Yang, 2013. Towards comprehensive provable data possession in cloud computing. Wuhan Univ. J. Nat. Sci., 18: 265-271.

Liu, C., R. Ranjan, X. Zhang, C. Yang and J. Chen, 2015. A Big Picture of Integrity Verification of Big Data in Cloud Computing. In: Handbook on Data Centers, Khan, S.U. and A.Y. Zomaya (Eds.). Springer, New York, ISBN: 978-1-4939-2091-4, pp: 631-645.

Liu, F., D. Gu and H. Lu, 2011. An improved dynamic provable data possession model. Proceedings of the IEEE International Conference on Cloud Computing and Intelligence Systems, September 15-17, 2011, Beijing, China, pp: 290-295.

Liu, H., P. Zhang and J. Lun, 2013. Public data integrity verification for secure cloud storage. J. Networks, 8: 373-380.

Mather, T., S. Kumaraswamy and S. Latif, 2009. Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance. 1st Edn., O'Reilly Media, Inc., USA., Pages: 338.

Mell, P. and T. Grance, 2011. The NIST Definition of Cloud Computing. NIST Special Publication, USA.

Menezes, A., P. van Oorschot and S. Vanstone, 1996. Handbook of Applied Cryptography. 1st Edn., CRC Press, UK.

Mukundan, R., S. Madria and M. Linderman, 2012. Replicated data integrity verification in cloud. IEEE Data Eng. Bull., 35: 55-64.

NCBI., 2014. Genome. Human Genome Project. http://www.ncbi.nlm.nih.gov/genome/.

Paillier, P., 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: Advances in Cryptology-EUROCRYPT '99, Stern, J. (Ed.). Springer-Verlag, Berlin Heidelberg, pp: 223-238.

Plank, J.S., 1997. A tutorial on reed-solomon coding for fault-tolerance in RAID-like systems. Software Pract. Exper., 27: 995-1012.

Ren, Y., J. Shen, Y. Zheng, J. Wang and H.C. Chao, 2015. Efficient data integrity auditing for storage security in mobile health cloud. Peer-to-Peer Network. Applic. 10.1007/s12083-015-0346-y

Shacham, H. and B. Waters, 2013. Compact proofs of retrievability. J. Cryptol., 26: 442-483.

Shokrollahi, A., 2006. Raptor codes. IEEE Trans. Inform. Theory, 52: 2551-2567.

Sookhak, M., A. Gani, H. Talebian, A. Akhunzada, S.U. Khan, R. Buyya and A.Y. Zomaya, 2015. Remote data auditing in cloud computing environments: A survey, taxonomy and open issues. ACM Comput. Surveys, Vol. 47. 10.1145/2764465

Sun, D.W., G.R. Chang, S. Gao, L.Z. Jin and X.W. Wang, 2012. Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. J. Comput. Sci. Technol., 27: 256-272.

Thakur, A.S. and P.K. Gupta, 2014. Framework to improve data integrity in multi cloud environment. Int. J. Comput. Applic., 87: 28-32.

Tripathi, A. and M.S. Jalil, 2013. Data access and integrity with authentication in hybrid cloud. Oriental Int. J. Innov. Eng. Res., 1: 30-33.

Wang, C., S.S.M. Chow, Q. Wang, K. Ren and W. Lou, 2013. Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput., 62: 362-375.

Wang, H. and Y. Zhang, 2014. On the knowledge soundness of a cooperative provable data possession scheme in multicloud storage. IEEE Trans. Parallel Distrib. Syst., 25: 264-267.

Wang, Q., C. Wang, K. Ren, W. Lou and J. Li, 2011. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst., 22: 847-859.

Xu, J. and E.C. Chang, 2011. Towards efficient provable data possession. IACR Cryptol. ePrint Arch., Vol. 2011.

Yang, K. and X. Jia, 2014. TSAS: Third-Party Storage Auditing Service. In: Security for Cloud Storage Systems, Yang, K. and X. Jia (Eds.). Springer, New York, ISBN: 978-1-4614-7872-0, pp: 7-37.

Yang, K. and X.H. Jia, 2012. Data storage auditing service in cloud computing: Challenges, methods and opportunities. World Wide Web, 15: 409-428.

Yuan, J. and S. Yu, 2013. Proofs of retrievability with public verifiability and constant communication cost in cloud. Proceedings of the International Workshop on Security in Cloud Computing, May 8-10, 2013, Hangzhou, China, pp: 19-26.

Zhang, Y. and M. Blanton, 2013. Efficient dynamic provable possession of remote data via balanced update trees. Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security, May 8-10, 2013, Hangzhou, China, pp: 183-194.

Zheng, Q. and S. Xu, 2012. Secure and efficient proof of storage with deduplication. Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy, February 7-9, 2012, San Antonio, TX., USA., pp: 1-12.

Zhu, Y., H. Hu, G.J. Ahn and M. Yu, 2012. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. Parallel Distrib. Syst., 23: 2231-2244.