Cluster Comput (2019) 22:S1113–S1123
https://doi.org/10.1007/s10586-017-1075-1

Secure provable data possession scheme with replication support in the cloud using Tweaks

S. Ahamed Ali[1] · **M. Ramakrishnan**[2]

Received: 10 April 2017 / Revised: 14 July 2017 / Accepted: 24 July 2017 / Published online: 17 August 2017

Abstract Cloud computing is an emerging model in which computing facilities are provided as a service and accessed using Internet hence organizations prefer data outsourcing to the cloud servers. Currently, organizations produce a large amount of data and expect increased availability, scalability and security. When data is migrated to cloud, security and availability of the data must be verified since the critical data of the organizations lies outside the data owner premises. To enhance the availability of the data the data owners prefer replicating data to more than one cloud servers. Thus there will be pre-defined SLAs between the cloud service provider (CSP) and the data owners that include payment of fees measured in terms of GB per month for data replication. We need to have a protocol which ensures that the cloud service provider is replicating the data storage based on the pre defined SLAs. In this paper we propose a protocol called "Secure Provable Data Possession scheme with Replication support in the Cloud using Tweaks" that prevents the CSP cheating the data owner by maintaining fewer replicas than the agreed one in the SLAs and also supports dynamic data operations. We illustrate the performance of our scheme with experimental analysis and prove that it performs better than the existing systems.

Keywords Cloud computing · Replication · Provable data possession · Data outsourcing · Tweaks

✉ S. Ahamed Ali
haiahamed@gmail.com

[1] Department of Information Technology, Velammal Engineering College, Anna University, Chennai, Tamil Nadu, India

[2] School of Information Technology, Madurai Kamaraj University, Madurai, Tamil Nadu, India

1 Introduction

The advent of emerging computing technologies such as service-oriented architecture and cloud computing has enabled us to perform business services more efficiently and effectively. Cloud storage enables users to remotely store their data and enjoy the on-demand high quality cloud applications without the burden of local hardware and software management. A cloud storage system is considered as a large-scale distributed storage system that consists of many independent storage servers. Cloud computing is a concept that treats the resources on the Internet as a unified entity, a cloud. Users just use services without being concerned about how computation is done and storage is managed. The cloud supports redundant, self recovering and scalable programming models that allows workloads to recover from many hardware and software failures.

Hence organizations and enterprises are showing huge interest in migrating their sensitive and important data to the cloud. More than half of all businesses utilize cloud services, with 95% of the general public using cloud in at least one form, according to a recent study from Soliant Consulting [20]. Cloud usage is expected to become even more widespread in 2017, with Soliant estimating that 36% of all data will be stored in the cloud by the end of 2017.

Cloud service (storage) providers such as Microsoft Azure, Amazon S3 invite data owners to store sensitive data on their storage servers. These providers attracts the data owners by offering various storage and security facilities. Currently, Microsoft's virtual private storage service [21], Amazon's virtual private cloud [22] etc., offers virtual private storage space for data storage. The data or the information that are stored in these types of cloud server's by the data owner is referred as data outsourcing and they are managed by the respective cloud service providers. So when the data or

application is being outsourced by an enterprise to the cloud service providers, their privacy becomes a very challenging task because the data owners (the cloud computing clients) have to trust the cloud providers on many fronts, especially on the availability of cloud service as well as data security. Therefore the Service Level Agreements (SLAs) forms an integral part of a data owner's first line of defense. Further security defenses are hosted along with the other user data on the cloud and are controlled by the service-providers. Thus other than the SLAs user can't hold a complete control over data security over his own critical data [15].

Data replication is a technique used by organizations to improve the availability of data. The explosive growth of data generated by organizations emphasizes the need for creating a new and advanced approach for data recovery and replication. Organizations need high speed data replication solution for enabling efficient backup and to reduce recovery time. However, because data capacity requirements are growing so rapidly, organizations are migrating to cloud based storage solution for their sensitive data backup and replication. Cloud data replication services provide cost savings and performance improvements. The cloud based data replication systems does not provide any evidence regarding the storage of multiple copies of data. The CSP can collude a single copy of data to make it look like they are storing many copies of the data, whereas in reality they only store a single copy.

Considering all the issues mentioned above we propose our model called "Secure Provable Data Possession scheme with Replication support in the Cloud using Tweaks (SPDPR-Tweaks)" that has the following properties

(1) It enhance the security of the outsourced data with a strong encryption scheme.
(2) Generates "n" unique replicas for a data block in a computationally efficient manner.
(3) Supports dynamic operations on the replicas.
(4) It supports provable data possession that allows data owner to store all the "n" replicas of a data block in a storage nodes provided by the CSP and verify the integrity of the replicas by challenge–response protocol.

The rest of this paper is organized as follows. In Sect. 2, we present literature survey. Section 3 presents our system model. Section 4 illustrates our integrity preserving storage model and Sect. 5 illustrates our new tweak based PDP-R scheme. In Sect. 6 detailed algorithms for dynamic data support are given. Section 7 explores the challenge–response paradigm and in Sect. 8 we conclude the paper.

2 Related works

Provable Data Possession (PDP) schemes were introduced by Ateniese et al. [3]. PDP is a technique that allows users, and the data owner, to check the integrity of their data without retrieving it. Ateniese et al. [3] have proposed a provable data possession (PDP) scheme that supports data appending operation which allows users to add new blocks at the end of the file. In their scheme the data owner before uploading the data to the cloud will split the File F into equal size blocks {b1,b2,b3,. . .,bn}, where n is the total number of blocks. The data owner will generate a unique tag for every outsourced block and store these tags in their trusted storage space. The data blocks are then outsourced in the cloud. During auditing phase the data owner send challenge message that contains the set of data blocks to be verified. The data owner will request the cloud service provider to generate tags for the specified data blocks. The data owner can then verify the response by using tags stored in the trusted storage space. If the returned tags match the stored ones, the data file F is not tampered. The absence of dynamic operations is the major pitfall in this method. To overcome this Erway et al. [9] proposed a scheme that supports dynamic operations on the outsourced blocks using rank-based authenticated skip lists which supports provable updates. This scheme is not efficient due to the use of authenticated skip list.

Xu et al. [12] proposed a new concept to prove the server data possession. In their algorithm, the client creates tags as polynomials and considers the file blocks as coefficients to polynomials. The proof procedure is based on polynomial commitment and uses evaluation in the exponential instead of bilinear maps. This idea is based on Lagrangian interpolation.

Replication is a method adopted in distributed environments like cloud in which data are stored across multiple sites in a network to enhance the availability and minimize the bandwidth consumption. The advantage of data replication is speeding up data access, reducing access latency and increasing data availability (Berl et al. [6], Long et al. [13]). Static replication strategies follow deterministic policies; therefore, the number of replicas and the host node is well defined and predetermined (Long et al. [13], Ghemawat et al. [10]) have proposed a Google File System (GFS) method to grab data replication. They have proposed a static method for replication which provides fast response, high availability, and high efficiency. The limitation of this algorithm is that a fixed replica number is used for all files which may not be the best solution for data.

Cidon et al. [8] have proposed a MinCopysets method which is a simple general purpose scalable replication scheme. MinCopysets has decoupled the mechanisms used for data distribution and durability. MinCopyset method incurs significant overhead on storage operations. Zeng and Veeravalli [19] have presented an optimal load balancing technique for large-scale cloud data centers that are composed of numerous Raw Data Server and many Meta Data Servers connected by arbitrary networks. In their algorithm

the number of replicas of each object is based on the demand rate for the object.

Dynamic strategies for data replication in cloud environments is proposed by, Bai et al. [5] they have used Response Time-Based Replica Management (RTRM) to create a replica for automatically enhancing the number of replicas based on the average response time. The major drawback of this method is low reliability, low load balancing and high replication cost. Boru et al. [7] have presented an energy-efficient data replication method in cloud data centers. In this replication approach, every data object is permanently stored at the Central Data Base (CentralDB) and depending on the access pattern, it is replicated in data center DB and Rack DBs.The problem with their approach is the cost of replication is high and updation of replica is very time consuming process. Gill and Singh [11] have presented an algorithm named Dynamic Cost-aware Re-replication and Re-balancing Strategy with the concept of knapsack problem to optimize the cost of replication. This algorithm has less consistency rate and response time.

Multiple-Replica PDP (MR-PDP) technique was proposed by Curtmola [14] which allows the data owner to verify the replicas of the outsourced file available with the CSP.It is the enhanced version of the method that was originally proposed by Ateniese et al. [3]. Pseudo-Random Function (PRF) was used to introduce some randomness in the replicas. Different PRF keys are used to generate multiple data copies. RSA signatures are used for tag generation. The disadvantages are the data is not encrypted, the size of RSA signatures is huge and finally it deals only with static data.

Barsoum et al. [4] proposed a scheme to support multiple replicas and data encryption. The data blocks in each copy are appended to file blocks before encryption. AES encryption scheme is used for data encryption and BLS signatures are used for tag generation. Dynamic data operations are supported with the help of Merkle Hash Trees. The disadvantage of this scheme is if a file has huge number of blocks, then number of nodes in MHT will be huge. The computation and communication overhead is the bottle neck problem to this algorithm performance. Considering all these limitations we present our SPDPR-Tweaks model in the following sections.

3 System model

Our proposed system model is presented in Fig. 1 and it consists of the following communication parties.

- The Cloud Service Provider (S): a third party who provides storage services to the data owners. The data owners can upload their data blocks to the storage space provide by S.

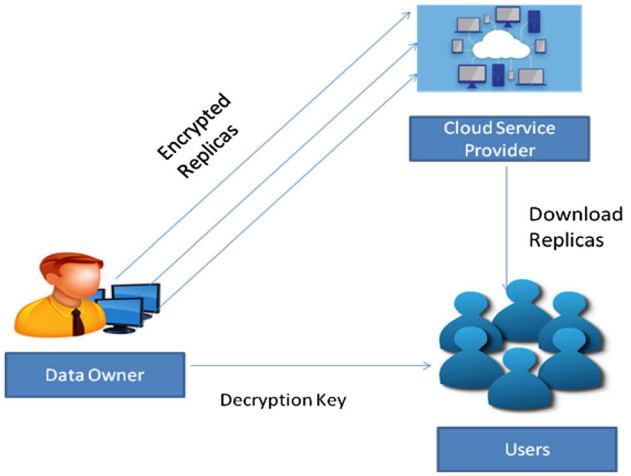

Fig. 1 System model for replication

- The data owner (D): is an enterprise or an individual who outsources the data in the cloud. The D will divide the file into fixed sized data blocks and generate multiple replicas for the data blocks
- The User (U): who has limited access rights to share or use the data block stored with the S.The User U will possess the valid decryption key to access all the encrypted data blocks.

To outsource a file to cloud service provider, the data owner encodes file F using some encoding technique such as erasure code to obtain file blocks Fi $= i$ 0, 1, 2, . . ., $n-1$. Thus only a fraction of blocks Fi's can recover the original file F using the decoding algorithm of the erasure code.

The data owner creates specific number of replicas for a block before doing encryption operation. Tweaks (discussed in Sect. 5) are used to generate unique replicas for every data block. Each replicated block is encrypted with a secret key. The encrypted replicas are outsourced to the storage nodes managed by the cloud service provider S. These storage nodes are maintained in a hierarchical tree based structure by the S and these storage nodes may be located at different locations.

Asymmetric key algorithms are used for encryption. All the authorized users are shared with the valid decryption key. In order to access the data blocks the authorized users request the CSP and download the encrypted blocks from the cloud and decrypt it using the decryption key. The data owner can verify the integrity of the replicas or request a trusted third party auditor to do it. To do so, the data owner can challenge the CSP to verify the integrity of the outsourced replicas for this operation we present a challenge–response scheme in Sect. 7 which is used to verify the integrity of the replicas.

4 Integrity preserving storage model with replication support

We use the Merkle hash tree (MHT) based storage model for providing security and preserving integrity for all replicas of the outsourced data blocks. MHTs are used to authenticate a set of n replicas with a constant size storage space [1,2]. A MHT for a set of "n" replicas R1,…,Rn, is a binary tree that has R1,. . .,Rn as leaves. An interior node of the tree with children C_L and C_R is the hash of the concatenation of its children (i.e., $h(C_L||C_R)$, for h a collision-resistant hash function. The value stored in the root of the tree thus depends on all the values stored in the leaves of the tree and can be used to authenticate all the leaf values. If the value of the root of the tree is stored by the data owner, then all the leaves of the tree can be authenticated by reading O(log(n)) hashes from the tree. We give example of a MHT that can store a maximum of eight replicas.

The binary tree structured merkle tree model shown in the Fig. 2 has many limitations. If the number of blocks in the file increases then the number of nodes in the MHT will also increase. This problem will become severe when replication of data blocks is essentially needed. The communication, computation overheads will increase and it reduce the performance of the system. To overcome this issue we propose a modified MHT structure, in which we replace the binary tree structure of the MHT to B+ tree structure. B+ Trees are special case of binary trees. It stores all the blocks in the leaf nodes. In a B+ tree there is an intermediary layer with nodes. They do not have actual blocks stored instead they are connected to the leaf node. Only the leaf node contains the blocks in some sorted order. In our approach Leaf nodes are connected through pointers to form a kind of linked list. This linkage helps in efficient traversal of the data blocks. In our approach the new block is always attached to the right hand side of the B+ tree. The B+ tree based MHT structure proposed in our approach has the following advantages.

- If file grows in its size, more blocks are needed to represent the file. Even in this case the performance of our approach remains the same. It does not degrade like the traditional binary tree structured merkle tree. This is because all the blocks are maintained at leaf node and all the nodes are at equi-distance from root. In addition, if there is any overflow, it automatically re-organizes the structure.
- Even though insertion and deletion are little complicated, it does not leads to any significant increase in computation time.

In the Fig. 3 h(bi,R) represents the hashed value for all replicas for the block "i" .The root node contains the hash value for the entire file. It may be noted that a File F is represented as a set of blocks $\{b_0,b_1,. . .,b_n\}$. Figure 4 represents the data access time when replication of data block is done. Comparison is done for the three variations of the MHT namely binary tree, general tree and the B+ tree for the same number of data blocks.

From the graph presented in Fig. 4 it is obvious that B+ tree based MHT approach performs better than the other two approaches. Similarly we have also studied the performance of B+ tree based MHT approach without replication support. Figure 5 shows the access time for data blocks without replication.

We carried out our experiments on text files. The file is divided into a set of data blocks and they are repre-

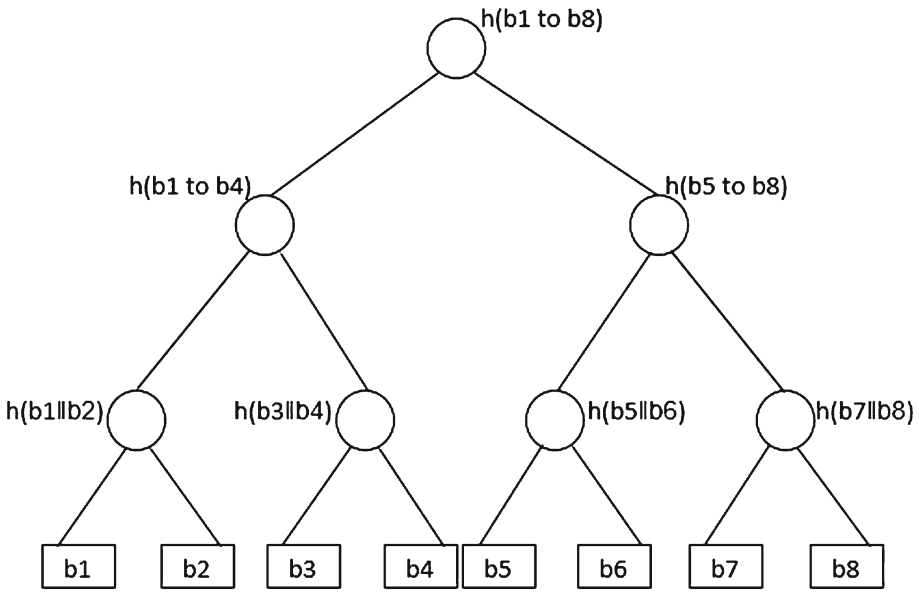

Fig. 2 Merkle tree with eight blocks

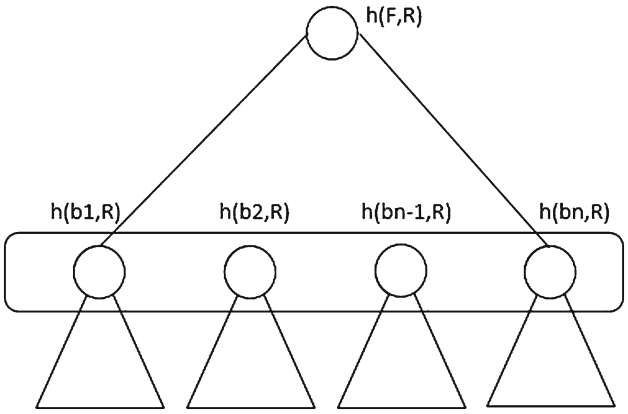

Fig. 3 B+ tree based Merkle tree structure

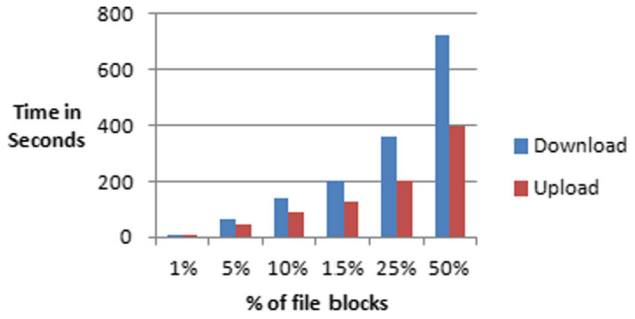

Fig. 6 Upload and download time

sented as a storage nodes in B+ tree based MHT. We plot the upload and download time for text files in Fig. 6. The block size was typically set as 2 kB for the files. The graph represents the upload and download time for a single data block.

5 Tweak based secure PDP-R scheme

The tweak based PDP-R scheme is used to enhance the security of the storage model discussed in the previous section and to generate unique replicas for a given data block. The tweak based Secure PDP-R algorithm is a three tuple algorithm that consists of three methods namely

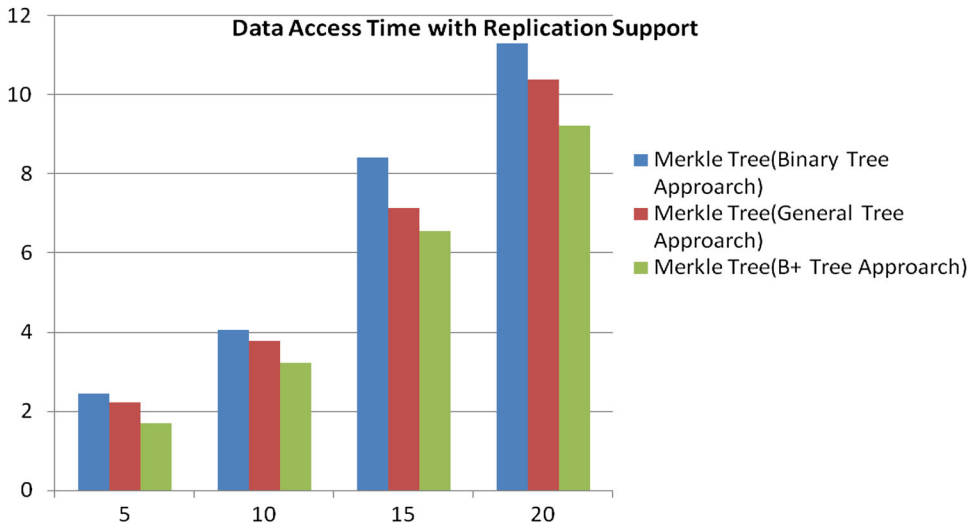

Fig. 4 Data access time with replication support

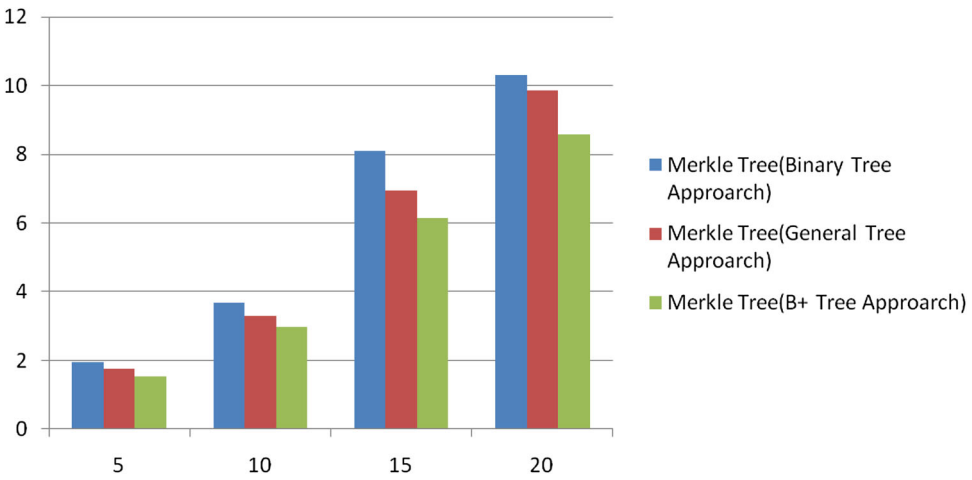

Fig. 5 Data access time without replication

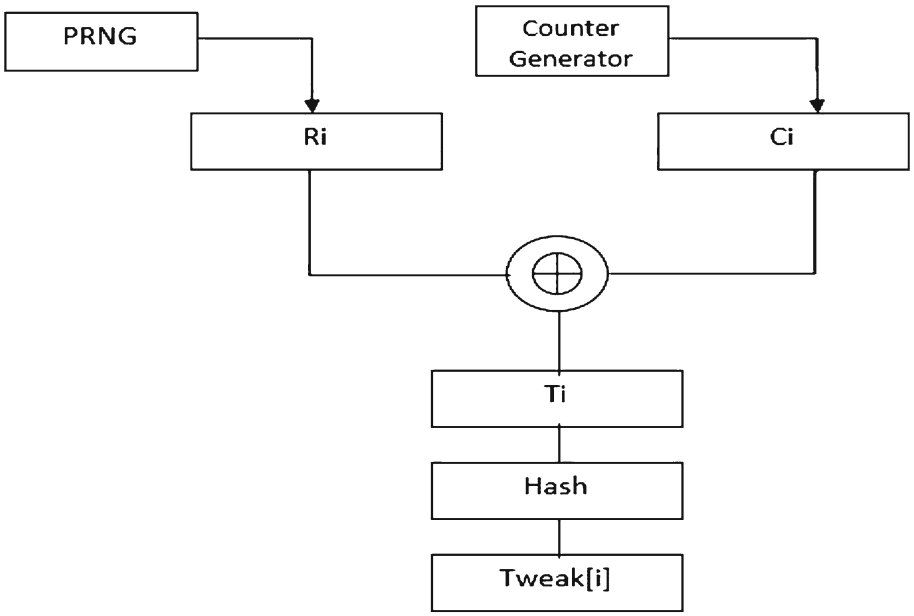

Fig. 7 Tweak generation

(a) ASYMKeyGen()—Used to generate public and private key pair.
(b) Tweak and replica generation method—It is a method that generates tweaks which is used for generating unique replicas.
(c) Encryption—It is a component used for encrypting the replicas for a given block.

(a) ASYMKeyGen()

It is an asymmetric key generation algorithm that generates two keys namely public (K_U) and private key (K_R). Private key is kept secret and the public key is shared with all trusted users in the system. This algorithm is executed by the data owner. We use elgammal based public key cryptosystem to generate private, public key pairs [16]. This cryptosystem is based on discrete logarithmic problem. The security of this cryptographic technique depends on difficulty in computing discrete logs in a large prime modules. The key generation part of the elgammal cryptosystem is illustrated as follows:

1. Generate a large prime p and a generator g of the multiplicative group Z_p^*.
2. Select a random integer a between 1,2,. . ., p−1.
3. Compute $(g^{a)}$mod p.
4. $K_U = \{p,g,g^a\}$.
5. $K_R = a$.

(b) Tweak and replica generation

(i) Tweak generation

Tweaks are generated to produce unique replicas for a given block. Tweaks provide strong evidence to ensure that the CSP is possessing all the replicated data copies as agreed upon the SLA and also all the replicated copies are intact. In addition tweaks enhance the security and produce good diffusion. Hence when different tweaks are used with the same plain text and the same key they produce different cipher text. To generate tweaks the data owner generates "n" random numbers by using a pseudo random number generator (PRNG). Let $\{ r_1, r_2, r_3 \ldots\ldots\ldots, r_n\} \in Z_N^*$ be the random numbers generated, these "n" random numbers are used for creating "n" replicas for a data block ("n" stands for the agreed number of replicas to be stored by the cloud). Each random number corresponds to creation of one replica for the given data block. Figure 7 presents the diagrammatic representation of tweak generation.

The following algorithm generates tweaks that are attached with the replicated blocks before encryption.

1. Generate "n" random numbers $\{ r_1, r_2, r_3 \ldots\ldots\ldots, r_n\} \in Z_N^*$ by using a PRG algorithm, here "n" stands for the required number of replicas to be generated.
2. Move all the random numbers generated to an array R[], such that R[i] corresponds to the ith random number.
3. Generate counters for all the "n" replicas for all the "m" blocks.

```
iv= { the value for iv will be given the data owner }
for i=1 to m do
for j = 1 to n do
     C[i][j]=iv
      iv++
 end for
 end for
```

4. Generate tweak for all the "n" replicas of all "m" blocks.

```
for i=1 to m do
 for i = 1 to n do
    let T[i][j]= C[i][j] ⊕ R[i]
    Tweak[i][j]=H(T[i][j]) , here H is a collision resistant one way hash function
end for
 end for.
```

(ii) Replica generation

This algorithm is run by the data owner to generate multiple replicas for the given data block. Before generating the replicas the entire file is preprocessed to produce a tag. For all the "m" blocks for a given file F a unique SEED is generated. SEED value will be computed only once i.e at the time of outsourcing the file F to the CSP. SEED is generated as follows:

$V_1 = E(K, b_1)$
for i = 2 to m do
$V_i = E(K, [b_i \oplus V_{i-1}])$
end for
SEED= V_i || len(F)

Once SEED is generated "n" replicas for all the "m" blocks for a given file can be generated by the following algorithm.

```
for i = 1 to m do
   for j=1 to n do
        replica[i][j]=SEED ⊕ tweak[i][j] ⊕ b_i
  end for
end for
```

(c) Encryption

Once replicas are generated, they are encrypted and outsourced to the cloud. An asymmetric key encryption algorithm is used for encryption and decryption. Encryption is done with the private key K_R and the decryption is done with the public key K_U. The encryption algorithm needs the public key K_U and the message block b_i as input.

1. Let bi be the message block to be encrypted.
2. Select a random integer "r" such that $r \in Z_n$.
3. Compute $\gamma = g^r \bmod p$.
4. Let $\alpha = b_i * (g^a)^r$.
5. The message block b_i can be encrypted using the equation $C_i = \{ \alpha, \gamma \}$.

The encrypted data blocks are uploaded to the CSP.

5.1 Performance of Tweak based secure PDP-R scheme

The performance of the proposed scheme is measured in terms of storage overhead and computation cost.

Storage overhead

Storage overhead is a measure use to identify the extra space required to store the information other than the outsourced data blocks. The storage overhead needed for the tweak based PDP scheme is much lesser when compared to the other tree based PDP schemes with dynamic data support. The tweak based PDP scheme needs to store the B+ tree based MHT for the outsourced blocks which results in the extra storage requirement on the CSP side. The B+ tree MHT structure has to be accessed for every dynamic data operations and also during the challenge–response (discussed in Sect. 7) time. Figure 8 represents the CSP storage overhead for both the traditional tree based PDP and the proposed tweak based PDP. It can be noted that in our proposed scheme the CSP storage overhead is independent of the number of data block this is because we use a B+ tree based merkle tree structure, in this structure all the data blocks are represented in the leaf nodes. These leaf nodes are equiv distant from the root node. Thus our B+ tree based MHT structure of height h and a minimum degree t always satisfies the equation $h <= \log_t (n + 1)/2$, where n is the number of replicas stored in the leaf nodes.

In all the traditional the tree based PDP approach the CSP storage overhead is linear with the number of data blocks. Hence the storage overhead increases as the number of block increases. In our approach by reducing the storage overhead we can reduce the payment of fees by the data owners calculated in GB/month for replicating the data storage.

Computation time

We now consider the computation time from the CSP side. The Computation time is measured in terms of the cryptographic and tree operations done on the data blocks. The computation time also includes the system setup time and challenge–response protocol execution time. The computation time also provides strong evidence that all the data blocks including the replicas are consistent and not corrupted. From the plot on the Fig. 9, it can be observed that the timing curve for tree based PDP grows more than our proposed tweak based PDP for the same number of data blocks .This is because the computation expression in the tree based PDPs contains more terms than our proposed approach.

6 Algorithms for dynamic data operations support

These algorithms are run by the data owner to perform any operation on the outsourced data blocks. The data owner sends the Update request to the cloud service provider with the parameters <FileId,BlockId>. The output of this algorithm would be the updated version of the block in the file.

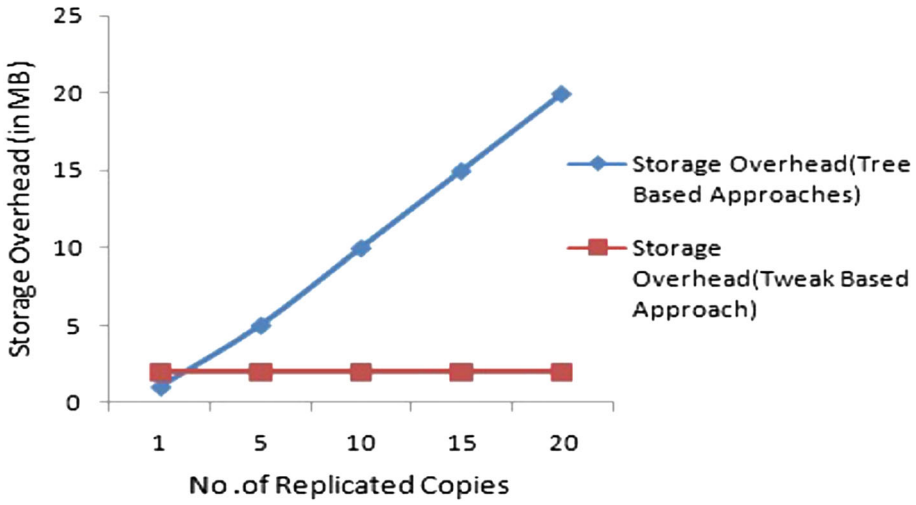

Fig. 8 Storage overhead versus no. of replicated copies

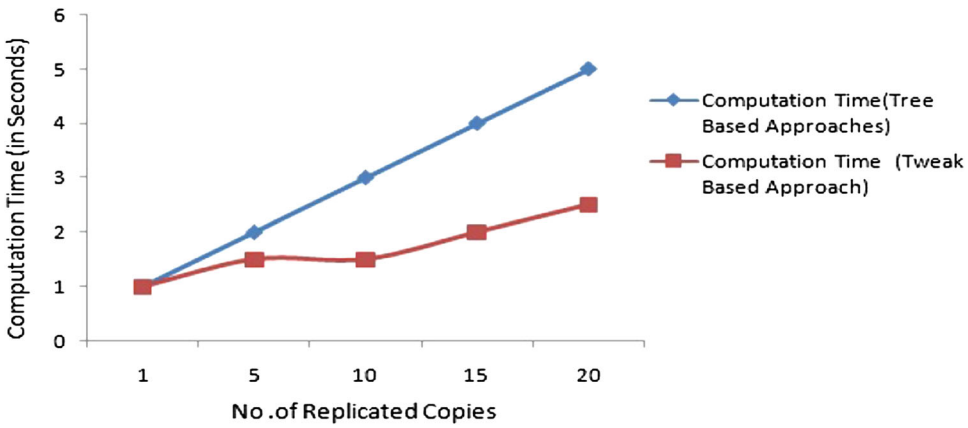

Fig. 9 Computation time versus no. of replicated copies

Insertion operation

1. When the new block is inserted, Tweak is computed and attached to the new block.
2. Replica generation algorithm is executed for generation of 'n' replicas for the new block.
3. Replicas are encrypted using the elgammal based encryption scheme.
4. The new block is inserted after the last block of the file.

Modification operation

1. Compute $\Delta b_i = b_i' - b_i$ where b_i' stands for the modified block b_i.
2. Encrypt Δb_i using elgammal based encryption method.
3. Compute $E(b_i)^* E(\Delta b_i)$ (by homomorphic property of encryption).

Deletion operation

When one block is deleted, indices of all subsequent blocks are moved one step forward thus maintaining the B+ tree structure.

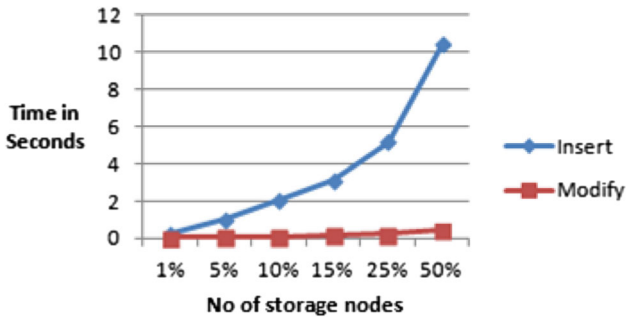

Fig. 10 Block operations

To study the performance of the dynamic data operations on the outsourced blocks we conducted experiments on multiple blocks. These experiments are executed from the data owner side. The update operation includes block insert, modify operations in addition to creation of tweaks. We ran the experiments for block update on a 1 MB file with a file block size of 128 bytes. The experiments are run by inserting and modifying 2 to 50% number of file blocks. The results are depicted in Fig. 10.

Table 1 Challenge–response scheme between the data owner and the CSP

S/N	Data owner	Cloud service provider
1	Generates a random key r and s $\in_R Z_N$	
2	Computes $g_s = g^s \bmod N$	
3	send r, g_s to the remote cloud server	
4		Generates random coefficients $\{aj = PRF(j)\} 1 \leq j \leq m$
5		Computes $R = (g_s) . \sum m$ $j=1$ aj .bj mod N
6		send R to the data owner
7	Generates a set of random coefficients $\{a_j = f_r(j)\}_{1 \leq j \leq m}$	
8	Computes $(T^{aj} \bmod N) \bmod N$	
9	Computes $R' = P^s \bmod N$	
10	Checks $R' ?= R$	

7 Challenge–response protocol design

Challenge–response is a protocol used by the data owner to verify the integrity of the outsourced blocks in the cloud [17,18]. Data owner has two options for such verification namely (a) Deterministic (b) probabilistic. In the first approach the data owner challenge the CSP to verify the integrity of the entire outsourced block including the replicas. In later case only a few blocks from all the copies are used for verification. In our approach we use the second method of integrity verification. In probabilistic approach we use only a fraction of outsourced blocks for integrity verification and it accounts for the entire block verification. To choose the blocks for integrity verification we use a pseudo random number generator which generates random number that corresponds to the block number for which the integrity to be verified. The challenge–response protocol has a set up phase in which public parameters are shared between the data owner and CSP these parameters are used during integrity verification. During this phase the data owner computes tweaks T_j for each block using the tweak generation process.

During the challenge phase the data owner generate a random number that corresponds to the block number for which the integrity has to be verified. In our algorithm the data owner generate a set of random co efficient to challenge the CSP, this property ensures that CSP cannot cheat the data owner by simply sending the response (the same challenge message) without doing any computation on the data stored in CSP side. The challenge and verification between the data owner and the CSP is based on the quadratic residues over the multiplicative cyclic group over modulo N with a generator g. Thus $s \in_R Z_N$ is kept secret and the generator g and r are kept public. In Table 1 we present the challenge–response mechanism in detail.

Setup

1. Compute N = pq is the RSA modulus (p and q are prime numbers).
2. Let g is a generator of Q_{RN} (QRN is the set of quadratic residues modulo N).
3. Public key pk = (N, g) and private key sk = (p, q).
4. f is a pesudo-random function.
5. File F = {b1, b2,…, bm}.
6. Data owner generates tweak Tj for each block bj.
7. The tweaks are stored on the owner side and the file is sent to the remote server.

We present the performance of our challenge–response scheme with other PDP schemes in the Table 2. The performance comparison is made for a file which is split into “m” number of blocks and with “n” number of challenges

8 Conclusion

Cloud computing is a technology that is continuously growing, and it is expected to successfully change the way we utilize information and communication technology resources. It also raises the demand for trust and security in cloud enabled

Table 2 Performance comparison

Performance parameters	Basic PDP scheme	PDP schemes with remote integrity checks	PDP Schemes at untrusted sources	Our scheme
Data owner computation overhead	Linear with the size of the file	O(m)	O(m)	O(1)
CSP computation overhead	Linear with the size of the file	O(n)	O(n)	o(1)
Data owner storage overhead	NO	NO	NO	NO
CSP storage overhead	O(m)	O(m)	O(m)	O(m)
No of challenges that can be made	Unbounded	Unbounded	Unbounded	Unbounded
Probabilistic/deterministic approach	Deterministic	Both	Probabilistic	Probabilistic

technologies. Hence security will become more important and will be a decision criterion for enterprises moving services into cloud computing technology. In this paper we listed various security concerns that may arise because of migrating the sensitive data to the cloud by organizations. Generally a data owner may request the cloud service provider to replicate some or all of their sensitive data to increase the availability and to enhance the performance. In the current scenario sufficient protocols are not there to ensure that the cloud service provider is maintaining the required number of replicas based on the SLAs. To overcome such issues we have proposed our "Secure Provable Data Possession scheme with Replication support using Tweaks" scheme. The tweaks generated in our approach helps to generate unique replicas for a block. The proposed encryption method strengthens our approach. We have also suggested a probabilistic challenge–response scheme to verify the integrity of the blocks. Extensive analysis shows that our scheme is provable secure, and the performance evaluation shows that our scheme is better than the existing protocols.

References

1. Ali, A.S., Ramakrishnan, M.: Security tag and hierarchical tree based dynamic key management technique for ensuring storage security to the data outsourced in the cloud. Int. J. Print. Packag. Allied Sci. **4**(2), 1320–1329 (2016)
2. Ali, S., Ramakrishnan, M.: Cryptographic Tree and Log file based secret sharing key management approach for securing outsourced data in the cloud. Asian J. Inf. Technol. **15**(20), 3973–3979 (2017)
3. Ateniese, G., et al.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM conference on Computer and communications security. Acm (2007)
4. Barsoum, A.F., Hasan, M.A.: Provable Possession and Replication of Data over Cloud Servers. Centre For Applied Cryptographic Research (CACR), University of Waterloo, Report 2010/32. http://www.cacr.math.uwaterloo.ca/techreports/2010/cacr2010-32.pdf (2010)
5. Bai, X., Jin, H., Liao, X., Shi, X., Shao, Z.: RTRM: a response time-based replica management strategy for cloud storage system. In: Proceedings of the grid and pervasive computing, pp. 124–133. Springer (2013)
6. Berl, A., Gelenb, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M.Q., Pentikousis, K.: Energy-efficient cloud computing. Comput. J. **53**, 1045–1051 (2010)
7. Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., Zomaya, A.Y.: Energy-efficient data replication in cloud computing datacenters. In: Paper presented at the Globecom Workshops (GC Wkshps), 2013 IEEE (2013)
8. Cidon, A., Stutsman, R., Rumble, S., Katti, S., Ousterhout, J., Rosenblum, M.: MinCopysets: derandomizing replication in cloud storage. In: Paper presented at the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI)
9. Erway, C., et al.: Dynamic provable data possession. In: Proceedings of the 16th ACM conference on Computer and communications security. Acm (2009)
10. Sanjay, G., Howard, G.: Leung Shun-Tak. The Google file system. In: Paper presented at the ACM SIGOPS operating systems review (2003)
11. Gill, N.K., Singh, S.: Dynamic cost-aware re-replication and rebalancing strategy in cloud system. In: Paper presented at the Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014 (2015)
12. Xu, J., Chang, E.: Towards efficient proofs of retrievability. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security, ASIACCS '12, New York (2012)
13. Sai-Qin, L., Yue-Long, Z., Wei, C.: MORM: a multi-objective optimized replication management strategy for cloud storage cluster. J. Syst. Arch. **60**(2), 234–44 (2014)
14. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: MR-PDP: Multiple- Replica Provable Data Possession. In: 28th IEEE ICDCS, 2008, pp. 411420 (2008)
15. Seny, K.: Cryptographic cloud storage. In Financial Cryptography workshops, pp. 136–149 (2010)
16. Taher, E.G.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Proceedings of CRYPTO 84 on Advances in cryptology, pp. 10–18, New York. Springer, New York (1985)
17. Wang, Q.: Enabling public verifiability and data dynamics for storage security in cloud computing. Comput. Secur. ESORICS **2009**, 335–370 (2009)
18. Wang, Q., et al.: Enabling public auditability and data dynamics for storage security in cloud computing. In: IEEE Transactions on Parallel and Distributed Systems, pp. 847–859, 22.5 (2011)
19. Zeng, Z., Bharadwaj, V.: Optimal metadata replications and request balancing strategy on cloud data centers. J Parallel Distrib. Comput. **74**(10), 2934–40 (2014)
20. http://www.soliantconsulting.com
21. https://www.nist.gov
22. https://aws.amazon.com/vpc

S. Ahamed Ali has completed his B.E. degree in Computer Science and Engineering from Bharadhidasan University and M.E. degree in Computer Science and Engineering from Anna University currently he is pursuing his Ph.D. degree from Anna University in the research area of Cloud Computing. He has a total of 15 years of experience in Teaching and Industry. He has published many papers in National and International Journals. He has also authored a book on OOPS and Java. At present he is working as an Assistant Professor in the Department of Information Technology at Velammal Engineering College, Chennai.

M. Ramakrishnan has obtained his Ph.D. degree in Computer Science and Engineering from Anna University. He has more than 25 years of experience in Teaching profession. He has authored several books and published many papers in International and National Journals. He is also the Associate Editor in International Journal of computational Intelligence Technologies. He has guided many Ph.D. scholars from various universities. He is a chief investigator in many funded projects. Currently he is the Professor and Chairperson in the School of Information Technology, Madurai Kamaraj University, Madurai.