Journal of Universal Computer Science, vol. 24, no. 5 (2018), 603-621
submitted: 23/1/18, accepted: 28/4/18, appeared: 28/5/18 © J.UCS

Efficient Privacy-Preserving Remote Data Possession Checking Protocol for Critical Information Infrastructure

Loganathan Yamuna Devi
(Department of Computer Applications, Coimbatore Institute of Technology, Coimbatore, India
yamunalogan@gmail.com)

Kaliannan Thilagavathy
(Department of Physics, Coimbatore Institute of Technology, Coimbatore, India
thilagavathy63@gmail.com)

Abstract: Critical infrastructure systems like finance, energy, transport, and healthcare provide very vital and crucial services to the government sectors, private organizations and general public. These services need reliable and scalable infrastructure to support their dynamically growing needs. Cloud computing provides such a reliable and scalable, as well as cost-effective infrastructure to support the ever increasing needs of those critical infrastructure providers. The critical sectors like healthcare, finance, energy and many other government and private organizations as well as individuals are moving their data to cloud storages. While enjoying the benefits of cloud, the users also face the challenges of data security and privacy. The distributed nature of cloud challenges the traditional form of securing the information in almost all domains of the society, and hence a new set of rules for securing information need to be defined. New avenues are to be explored to defend against the threats posed towards sensitive data hosted in cloud. In this paper an efficient, privacy-preserving Remote Data Possession Checking protocol is proposed for verifying the integrity of data stored in cloud. This Remote Data Possession Checking (RDPC) protocol uses random sampling of data blocks in order to eliminate the need for accessing the entire data file each time it is verified. Also an exhaustive security and performance analysis of the proposed protocol is performed under various parameter values. The experimental results show that the proposed protocol is efficient in terms of performance when compared to many existing schemes, and require only minimum number of data blocks to detect the corrupted blocks with high probability.

Keywords: Critical Infrastructure Security, Cloud Computing, Remote Data Possession Checking, Data Integrity, Privacy-Preserving, Spot Checking, Random Sampling.
Category: K.6.5

1 Introduction

Critical sectors like finance, energy, transport, and healthcare are vital for the economy and the society of a country. In order to manage the huge volumes of data generated, and to support enormous processing power needed to process those data, these services need reliable and scalable infrastructure. Cloud computing is the emerging paradigm which offers infrastructure, development, deployment and testing platforms, and software solutions as services on demand basis. Among the many services cloud computing provides, storage is a mainly used resource. The critical sectors like finance, energy, transport, healthcare, education, banking and some of the social networks like Facebook, LinkedIn, and some of the government sectors like Defence and R&D move their huge volume of data to cloud data centres because of the benefits offered by the cloud. Cloud Computing represents convergence of technologies such as high-speed broadband, large scale data centres, and flexible virtualization software. Working together as cloud computing, these forces are driving one of the most important global technology transformations impacting on many types of business, political, and social structures. According to recent statistics from Forrester Research, traditional technology models are replaced by this new cloud paradigm. From smart phones to tablet computers, the cloud has pervaded modern life. Because of its on-demand provisioning of virtualized resources, cloud has shifted the power of computing from large organizations toward individual consumers. The net effect of new cloud-based models is a decentralization of business and social structures. Organizations or individuals no longer need to acquire the resources, but can use through the cloud.

1.1 Motivation

The European Network and Information Security Agency (ENISA) [ENISA 2012], who has published the white paper, *Critical Cloud Computing: A CIIP Perspective on cloud computing services,* underlines the strengths of cloud computing, when it comes to dealing with natural disasters, regional power cuts and DDoS attacks. At the same time it highlights that the impact of cyber attacks could be very large, because of the concentration of resources. In information security, the concentration of IT resources is a 'double edged sword': On the one hand, large cloud providers can deploy state of the art security and business continuity measures and spread the associated costs across the customers. On the other hand, if an outage or a security breach occurs then the consequences could be big, affecting many citizens, many organizations, at once. The concentration of IT resources makes cloud computing services critical and relevant to look at from a CIIP perspective.

In healthcare sector, the data such as Patient Personal Information (PPI), medical data in the form of images, radiological reports, x-rays are maintained electronically. These electronic medical data are highly sensitive and private as well as posses substantial monetary value. Similarly in Financial Sectors such as Banks and Insurance agencies, the customer details, account details and transaction details are highly confidential. Breaching these highly sensitive, private, and confidential data results in loss of business and reputation and erode the confidence of customers and patients. Federal laws such as Health Insurance Portability and Accountability Act (HIPAA), Gramm-Leach-Biliey Act for Financial Companies, and Payment Card Industry Security Standard mandate that business and institutions protect the security and privacy of personal data.

In 2016, the EU passed the General Data Protection Regulation law (GDPR), which addresses the handling of personal data of EU citizens in two broad areas - the protection of personal data and the free movement of that data within the borders of the EU [EU GDPR 2016]. When the GDPR will come into efffect, organisations must be able to meet requirements to demonstrate personal data governance and prove they are keeping personal data private and secure. Organisations must be able to map personal data so they know what data resides where and how it is processed. Access rights to that data must also be put in place with audit trails to demonstrate compliance. As a privacy-by-design best practice, data should be encrypted and, where necessary, masked to avoid re-identification of data subjects. Organisations should take the sole responsibility of securing all personal data on their journey towards GDPR compliance. Regardless of where data is physically stored, the organisation remains data controller with ultimate responsibility for any breach of GDPR and therefore must have full view of any data in the cloud run by third-party data processors. Putting security capabilities in place for tracking and encryption of data is necessary – even if a third-party cloud data processor offers built-in security.

After the users and organizations move their data to cloud, they have no control over their data, since it is remotely administered. Either malicious users or misbehaving servers may alter or delete the data. They knowingly or unknowingly do so for monetary or other benefits, and claim that they poses the original, unaltered data to establish their reputation. Cloud service providers should schedule frequent audits and tests, by internal testers and auditors, and when relevant, by external testers and auditors. But it is hard for an external auditor to assess the security of a complex and continuously changing system, by performing an audit once per year. Cloud computing providers and government authorities should have a continuous program of monitoring, audits, tests and exercises in place [ENISA 2014]. Hence the cloud users, who have outsourced their data to cloud storage servers need some means of continuously verifying the integrity of their outsourced data. However, the data may be so large that the user simply asking for it to be downloaded every so often for integrity verification may not be feasible. This leads to the topic of *Proofs of Data Possession.* These are cryptographic protocols which enable a storage provider to prove to a client that certain files are still being stored, without the client needing to keep copies of all that has been stored [Ateniese et al. 2011]. Hence the objective of this paper is to design a secure and efficient Remote Data Possession Checking protocol (RDPC) with Random Sampling to verify the integrity of user's data stored in remote cloud storage without retrieving the whole data.

The organization of the paper is as follows: Section 2 presents the literature review, and section 3 elucidates the problem statement and the system model. Section 4 explains about the proposed system and the Challenge / Response protocol used in this system. Section 5 gives the Security and Performance analysis of the proposed protocol and section 6 concludes the paper.

2 Literature Review

The problem of verifying the correctness of outsourced data has been studied by many researchers. [Deswarte et al. 2003] initially provided solution to address this problem. They used RSA based hash functions to hash the entire file at every challenge. The drawback of this scheme is that the prover needs to access the entire file for each verification. It incurred huge communication overhead for the verifier. [Sebe et al. 2008] proposed an RSA based Remote Data Possession Checking (RDPC) protocol that allows an unlimited number of verifications. But using RSA is extremely slow and incurs storage overhead. [Jules and Kaliski 2007] introduced the concept of Proof of Retrievability (PoR) in which a client can retrieve a file stored previously at a remote server. This scheme uses special blocks called sentinels hidden randomly among regular file blocks in the data file. It can handle only limited number of verifications, and the use of sentinels increases the storage overhead considerably. [Erway et al. 2009] proposed a framework and a construction for dynamic provable data possession, but that is not privacy-preserving. [Ateniese et al. 2011] proposed two provably secure PDP schemes. Both schemes used RSA based homomorphic verifiable tags. In these schemes the server generates a proof of possession by randomly selecting set of file blocks. This random selection of file blocks eliminates the need for accessing the entire file for each verification. But the drawbacks of these schemes are that, they do not preserve the privacy of user's data and they are based on public key based cryptography which are computationally expensive when compared to symmetric key cryptography. The table 1 shows the key features of some of the existing schemes that are compared against our proposed scheme.

Scheme	**Metrics**				
	Cryptography based on	**Type of Guarantee**	**Public Verifiability**	**Privacy Preserving**	**Support for Sampling**
[Ateniese et al. 2011]	Public Key	Probabilistic	Yes	No	Yes
[Erway et al. 2009]	Public Key	Probabilistic	No	No	Yes
[Wang Q et al. 2011]	Public Key	Probabilistic	Yes	No	Yes
[Sebe et al. 2008]		Deterministic	No	No	No
[Zhu et al. 2010]		Probabilistic	Yes	Yes	Yes
[Hao et al. 2011]		Deterministic	Yes	Yes	No
[Chen 2012]	Symmetric Key	Probabilistic	No	No	Yes
[More and Chaudhari 2016]	Symmetric Key	Deterministic	Yes	No	No
[Singh and Pasupuleti 2016]		Probabilistic	Yes	No	Yes
[Saxena and Dey 2016]	Public Key	Probabilistic	Yes	Yes	Yes
Proposed Scheme	Symmetric Key	Probabilistic	No	Yes	Yes

Table 1: Comparision of the proposed protocol with the existing schemes

Most of the RDPC schemes in the literature survey [Wang C et al. 2011], [Wang Q et al. 2011], [Saxena and Dey 2016], [Singh and Pasupuleti 2016], [Feng and Long

2017] use public-key cryptography based RDPC which are computationally expensive, especially for massive data. Some of the schemes [Wang et al. 2012] are not privacy-preserving and the number of verifications are limited. [Chen 2012] has proposed a scheme for checking data possession in cloud storage using algebraic signatures. But using algebraic signature is not cryptographically secure, since it is easy to construct two strings that have the same algebraic signatures. [Hao et al. 2011] proposed a privacy-preserving remote data integrity checking protocol. But that scheme does not handle meta data information leak. The auditor can make use of the meta-data to learn information about the stored data. [Yu et al. 2014] developed a public-key based model to address this drawback. The scheme proposed by [More and Chaudhari 2016] computes tags for all encrypted file blocks and hence the computation and storage overhead is high. Some of the existing schemes like [Chen 2012], [More and Chaudhari 2016], [Singh and Pasupuleti 2016] fail to preserve the privacy of the outsourced data.

Our proposed protocol for cloud data integrity verification satisfies most of the requirements for remote data possession checking. It has several advantages over the existing schemes. Firstly, the proposed protocol uses symmetric-key cryptography which improves efficiency in terms of computation and storage. Second, it uses cryptographically secure, collision-resistant hash algorithm for data integrity verification. Third, it allows integrity verification without requiring the original data. It uses only the pre computed tags of the data and the response returned by the server to ensure the correctness of outsourced data. Fourth, it preserves the privacy of the outsourced data while at storage and during verification process by encrypting the data before it is outsourced to the cloud storage. This requirement is very essential for critical data such as patient's medical data. Finally, the use of sampling of data blocks for proof generation and verification is very efficient when large data need to be verified as in the case of cloud storage.

3 Problem Statement

3.1 Cloud Computing Use case in Healthcare Industry

The healthcare industry is shifting toward an information-centric care delivery model. Cloud computing provides an IT infrastructure that allows hospitals, medical practices, insurance companies, research facilities and other organizational entities in the healthcare ecosystem to leverage improved computing capabilities at lower initial capital expenditure [CSCC 2017]. In healthcare sector, the data such as Patient Personal Information (PPI), medical data in the form of images, radiological reports, x-rays are maintained electronically. These huge volumes of electronic medical data mandate the healthcare organizations to move their patient's medical data to third party storage such as cloud storage. Having just one copy of data opens up to vulnerabilities in the event of diaster. Backup as a Service (BaaS) is an approach to data backup and retention where the healthcare organization outsources their backup and recovery services to an online data backup cloud service provider [CSCC 2017]. Restoring backups from the cloud is fast and can help organizations avoid catastrophic data loss.

Healthcare data is specifically addressed in GDPR. Healthcare data is treated as personal data and requires the security and privacy protection that applies to all personal data [CSCC 2017]. If regulations and laws such as HIPAA for medical records in United States, GDPR in European Union must be enforced, the cloud provider must be able to prove their compliance. Many consumer requirements include adherence to legal regulations or industry standards. It is vital that the consumer be able to audit the provider's systems and procedures. An SLA should make it clear how and when these audits take place.

Motivated by the above scenario, we propose a secure and efficient Remote Data Possession Checking protocol (RDPC) with Random Sampling to audit correctness of personal data stored in cloud storage without retrieving the whole data.

3.2 System Model

The Remote Data Possession Checking scheme follows the Challenge / Response protocol in which the verifier (data owner / third party auditor) challenges the server to provide a *proof of data possession* for which the prover (cloud server) will have to respond. This response is verified by the verifier whether it meets the challenge, and hence the integrity of the outsourced data is verified [Sebe et al. 2008].

In order to meet the lightweight requirement of Remote Data Possession Checking, the technique of *Spot Checking* is used. In this technique, instead of auditing the entire data file, only randomly sampled file blocks are used for data integrity verification. This is because every time downloading the entire data is impractical and introduces unnecessary overhead to the verifier and the storage server. Though this is only a probabilistic verification, spot checking assures high probability of server misbehaviour detection with minimum number of file blocks sampled for verification.

The RDPC scheme normally involves two phases: (i) File preprocessing and Challenge computation, and (ii) Verification process. These phases can be implemented using a collection of four algorithms (File Setup, Tag Generation, Proof Generation, and Proof Verification).

(i) File Preprocessing and Challenge computation:

In this phase, the data owner preprocess the data file and computes verifiable tags to be used during the verification phase. During preprocessing, the data owner may alter the data file such as encrypting, encoding, or expanding the file, or may include additional metadata. He then outsource the data file to the remote server for long term storage and sends the metadata to the auditor. These processes can be implemented using the algorithms *File Setup* and *Tag Generation* and are shown in Figure 1.

In *Tag Generation,* 'c' data blocks are randomly sampled and tags are generated for the sampled set of data blocks. This reduces the need for accessing the entire file for verification. The positions of the data blocks to be sampled for tag computation is given by the following Equation (1).

$$I_j = \sigma_{k_i}(r2 + j) \qquad (1)$$

Each tag is then generated using the sampled blocks as given by Equation (2).

$$Tag_i = \sum_{j=1}^{c} hash\ (F_{I_j}) \qquad (2)$$

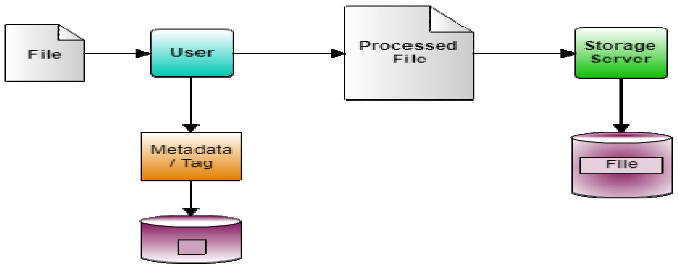

Figure 1: File Preprocessing

(ii) Verification Process:

At periodical intervals, the verifier posses a random challenge to the storage server by providing some metadata. The storage server meets the challenge by computing the tag (*Tag'$_i$)* for the challenged file blocks using the Equations (1) and (2) mentioned above, and return the response to the verifier. The verifier compares this response with the already computed tags. If *Tag'$_i$* = *Tag$_i$*, then the server has passed the challenge. Otherwise, the verifier concludes that the data is modified after it is stored in the remote server. These processes can be implemented using the algorithms *Proof Generation* and *Proof Verification* as shown in Figure 2.

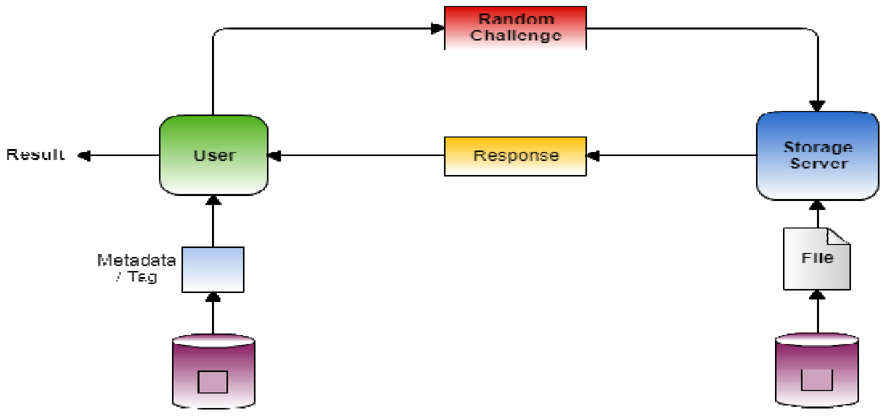

Figure 2: Verification Process

4 Proposed System

An efficient Remote Data Possession Checking Protocol with Random Sampling is proposed for secure cloud storage to verify the integrity of the outsourced data by challenging the cloud server. The proposed scheme is advantageous over many of the existing schemes like [Wang et al. 2012], [Wang C et al. 2011], [Wang Q et al. 2011], which use public key encryption. The proposed scheme uses symmetric key encryption, which is efficient in terms of computation and storage. Lanxiang Chen [Chen 2012] has proposed a scheme for checking data possession in cloud storage using algebraic signatures. But that scheme is not cryptographically secure, since it is easy to construct two strings that have the same algebraic signatures. The proposed protocol uses cryptographically secure hash algorithms for tag generation.

4.1 Desirable Properties

The proposed protocol is both *lightweight* and *efficient*. It is lightweight since it does not overburden either the verifier or the prover. This can be achieved by *spot checking,* in which the verifier samples small number of data blocks for integrity verification instead of taking the entire file. Efficiency of RDPC protocol with random sampling can be assessed based on:

- Communication Overhead - Initial transfer of data blocks to the server by the data owner, transfer of challenge by the verifier and response by the server.
- Computational Overhead - Computation of tokens by the data owner before uploading the data to the server, computation of response by the server, and verification of response by the auditor.
- Storage Overhead - Storage space needed for storing the file blocks and the tokens. The data owner computes tokens on the encrypted file blocks and uploads the file blocks to the storage server and sends the pre computed tags and metadata needed for challenge to the auditor.

4.2 Notations

Notations
f : $\{0,1\}^k$ x $\{0,1\}^l$ -> $\{0,1\}^l$
σ : $\{0,1\}^k$ x {1,2,…n} -> {1,2,…n}
where f( ) is a pseudo-random function (PRF), and σ( ) is a pseudo-random permutation (PRP)
k : the master key
k_e : the encryption key
E_{ke}( ) : Encryption Algorithm
r1, r2 : random numbers
F : original data file
$f_1, f_2, \ldots f_n$: data blocks of the file F
$F_1, F_2, \ldots F_n$: encrypted data blocks of the file F
$Tag_1, Tag_2, \ldots Tag_n$: tags of the data blocks

4.3 The Challenge / Response Protocol for Remote Data Possession Checking

The proposed RDPC protocol with random sampling uses the Challenge / Response protocol for verifying the integrity of remotely stored data. This protocol involves the following five phases: (i) File Setup, (ii) Tag Generation, (iii) Challenge, (iv) Proof Generation, and (v) Proof Verification as shown in Figure 3.

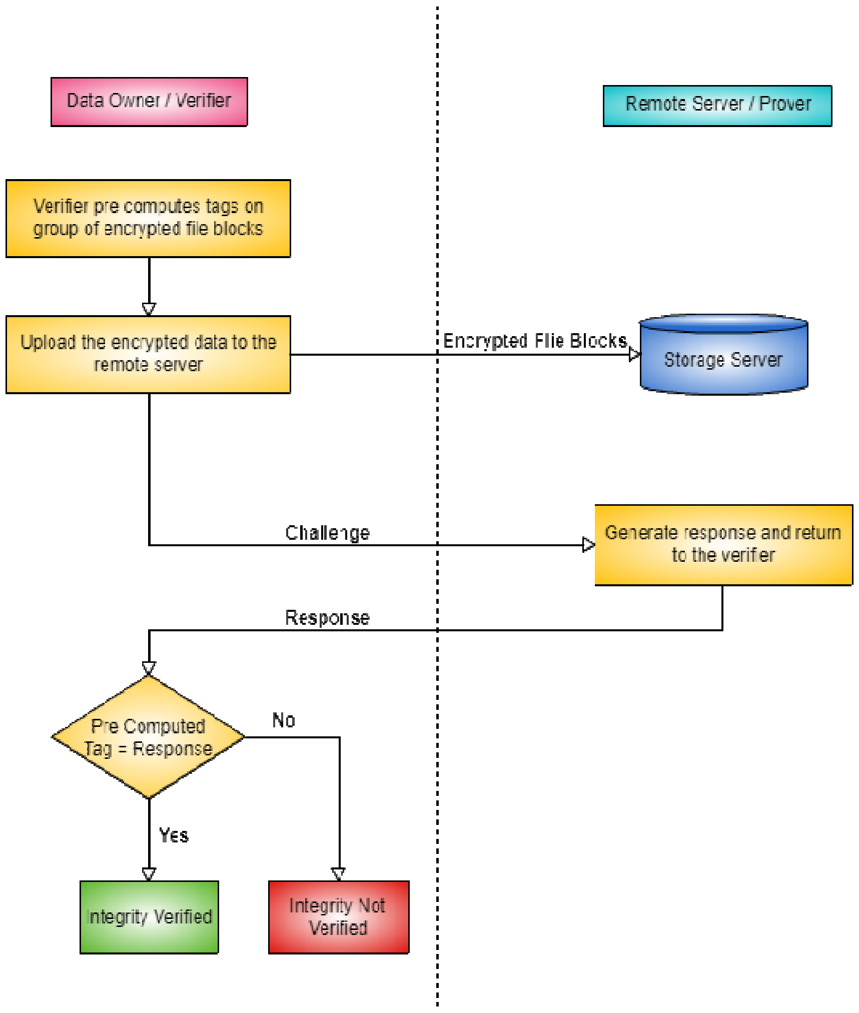
Data Owner / Verifier
Remote Server / Prover
Verifier pre computes tags on group of encrypted file blocks
Upload the encrypted data to the remote server
Encrypted File Blocks
Storage Server
Challenge
Generate response and return to the verifier
Response
Pre Computed Tag = Response
No
Yes
Integrity Verified
Integrity Not Verified

Figure 3: Challenge / Response Protocol for Data Integrity Verification

(i) File Setup :

During file setup, the data owner splits the original file F into 'n' fixed size blocks f_1, f_2, … f_n, generates the master key 'k' and encryption key 'k_e', and generates two random numbers r1 and r2. He then encrypts each file block f_i using Symmetric key encryption algorithm $F_i = E_{ke}(f_i)$ using the Equation (3). This step is done to ensure the privacy-preserving property of the protocol, ie, the original file contents are not revealed to the prover during proof generation.

That is, $F = f_1, f_2, \ldots. f_n$ where $n \geq 1$.

$$E_{k_n}(F) = \sum_{i=1}^{n} E_{k_n}(f_i) \tag{3}$$

Algorithm 1 : File Setup

```
1 : procedure SplitFile (File F)
2 :     while (all the bytes in F are read)
3 :         create new file $f_i$;
4 :     end while
5 :     generate master key K;
6 :     generate encryption key Ke
7 :     for (i from 1 to n )
8 :         $F_i$ = encrypt ($f_i$, $key_i$) ;
9 :     end for
10: end procedure
```

(ii) Tag Generation:

This phase involves computing tags for encrypted file blocks which will be used for verifying the proof returned by the server. Here the concept of *spot checking* is used in which tags are computed for randomly selected group of file blocks. Data owner computes *'m'* tags $Tag_1, Tag_2, \ldots Tag_m$ where 'm' is the number of verifications. For each tag Tag_i, *'c'* random file blocks are chosen and tag is computed for that group of blocks.

Algorithm 2 : Tag Generation

```
1 : procedure TagGen
2 :   for $0 < i < m$
          $k_i = f_k(r_1 + i)$;
          $s = 0$;
3 :       for $0 < j < c$
              $I_j = \sigma_{ki}(r2+j)$;
4 :           $Tag_i = s + hash(F_{Ij})$ ;
5 :       end for
6 :   end for
7 :   store all $Tag_i$ ' s locally
8:    upload all $F_i$ to Cloud Server
9 : end procedure
```

(iii) Challenge:

The Data Owner / Verifier, challenges the Sever with necessary metadata. For the i^{th} challenge, Verifier computes $k_i = f_k(r_1 + i)$, and sends $(r2, k_i)$ to the storage server.

(iv) Proof Generation:

The storage server / prover, when challenged by the data owner / verifier, computes the locations of the blocks for which the tag is to be computed, and computes Tag'_i for the group of blocks and sends the response to the verifier.

Algorithm 3 : Proof Generation

```
1 : procedure ProofGen
2 :         s = 0;
3 :         for 0 < j < c
                  $I_j = \sigma_{ki}(r2+j)$;
4 :               $Tag'_i = s + hash(F_{Ij})$ ;
5 :         end for
6 :         return $Tag'_i$ ;
8 : end procedure
```

(v) Proof Verification:

Upon receiving the response from the storage server, the verifier compares the tag (Tag'_i) returned by the server with pre computed tag (Tag_i) for that group of blocks. If $Tag'_i = Tag_i$, then the integrity of the data stored at the remote cloud server is verified.

The proposed protocol is a generic technique that can be applied to any auditing scheme regardless of the specific algorithms used. The protocol is both lightweight and efficient and preserves the privacy of users data.

5 Security and Performance Analysis

5.1 Security Analysis

The proposed protocol preserves the privacy of user's data by encrypting the file blocks using symmetric key encryption algorithm before the file blocks are outsourced to the cloud server. This protocol also uses cryptographically secure hash function for tag generation as well as proof generation.

Parameters
n : Total number of file blocks
x : Number of corrupted blocks
c : Number of queried blocks
Px : Probability that at least one of the blocks picked by client (C) matches one of the blocks corrupted by server (S).
P_T : Detection probability during the audit period 'T'
f : Audit frequency - number of occurrences of an audit event per unit time

The proposed protocol provides only probabilistic proof of data possession since the verifier asks for the proof of only randomly selected group of file blocks. Deterministic proofs can be provided only when all the file blocks stored in cloud server are used for proof generation. Though this is only a probabilistic verification, spot checking assures high probability of server misbehaviour detection with minimum number of file blocks sampled for each verification.The strength of the proposed protocol against server misbehaviour detection can be evaluated in two aspects: the error detection probability, and the frequency of periodic verification.

(i) Error Detection Probability:

It is assumed that the data owner / client (C) stores *n* - block file in the remote server (S). Intentionally or unintentionally the server may corrupt *x* blocks out of the *n* blocks. The Client C periodically performs integrity checking by randomly selecting *c* blocks out of the *n* file blocks.

Let X be a discrete random variable defined as the number of blocks queried by C that match the blocks corrupted by S. Then P_X, the probability that at least one of the blocks queried by C matches one of the blocks corrupted by S, is computed using the Equation (4).

$$P_X = P\{X \geq 1\} = 1 - P\{X = 0\}$$

$$1 - \frac{n-x}{n} \cdot \frac{n-1-x}{n-1} \cdot \frac{n-2-x}{n-2} \cdots \frac{n-c+1-x}{n-c+1}$$

Since $$\frac{n-i-x}{n-i} \geq \frac{n-i-1-x}{n-i-1}$$

it follows that

$$1 - \left(\frac{n-x}{n}\right)^c \leq P_X \leq 1 - \left(\frac{n-c+1-x}{n-c+1}\right)^c \qquad (4)$$

P_X indicates the probability that if server S corrupts *x* blocks of the file, then client C detects server misbehaviour by asking proof for *c* blocks. In the scheme, the number of verification and the number of blocks required for each challenge can be varied according to user's requirement. For more sensitive data, proof for large number of blocks can be asked in order to detect the data corruption with high probability. For less sensitive data, proof for only less number of data blocks is sufficient.

The Figure 4 shows the probability of server misbehaviour detection where client C asks proof for 5% of total file blocks (ie, c = 5 % of n) under various values of n and x. The figure shows that if x = 1% of n, then the Client asks proof for 460 and 300 blocks to achieve P_X of at least 99% and 95% respectively. It is also shown from the figure that this probability can be improved by asking proof for more number of blocks, and also the number of queried blocks for each challenge decreases with increasing number of corrupted blocks.

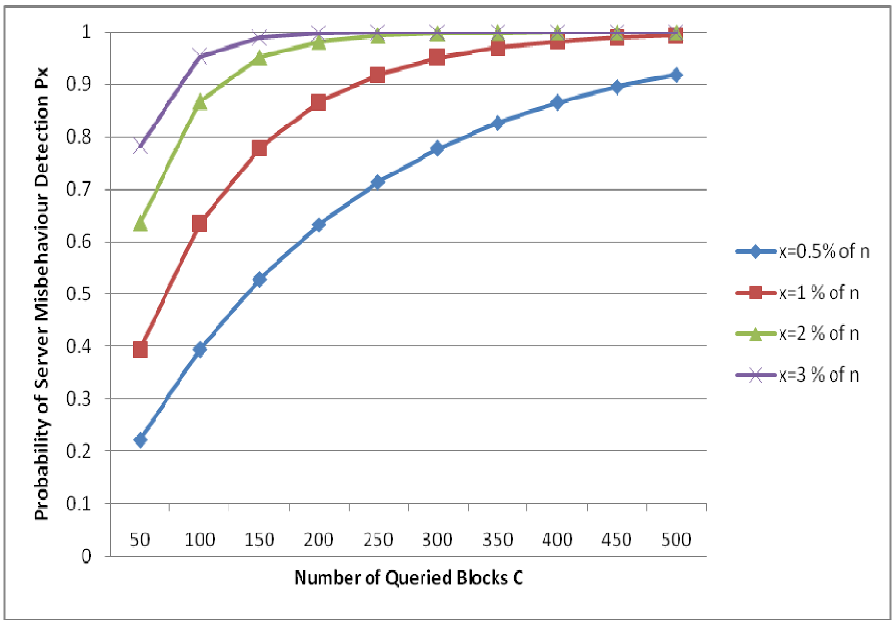

Figure 4: Probability of Server Misbehaviour Detection for 1% Corrupted Data

Total file blocks Vs Ratio of Queried blocks:

When the number of corrupted blocks (x) is 1% of the total number of file blocks, the ratio of queried blocks to the total number of file blocks is given by the Equation (5).

$$\text{Ratio } w = \frac{c}{n} = \frac{\log(1-P_X)}{n.\log(1-r)} \qquad (5)$$

where *r* is x / n.

To clearly represent this ratio, Figure 5 plots w for different values of n, x, and P_X. This ratio changes for different detection probabilities under 1% corrupted blocks. This ratio is higher for smaller file sizes and tend to approach a constant lower value for larger files. Hence for very large files in the order of gigabytes and terabytes, only a small fraction of file blocks is sufficient to verify the correctness of remotely stored data.

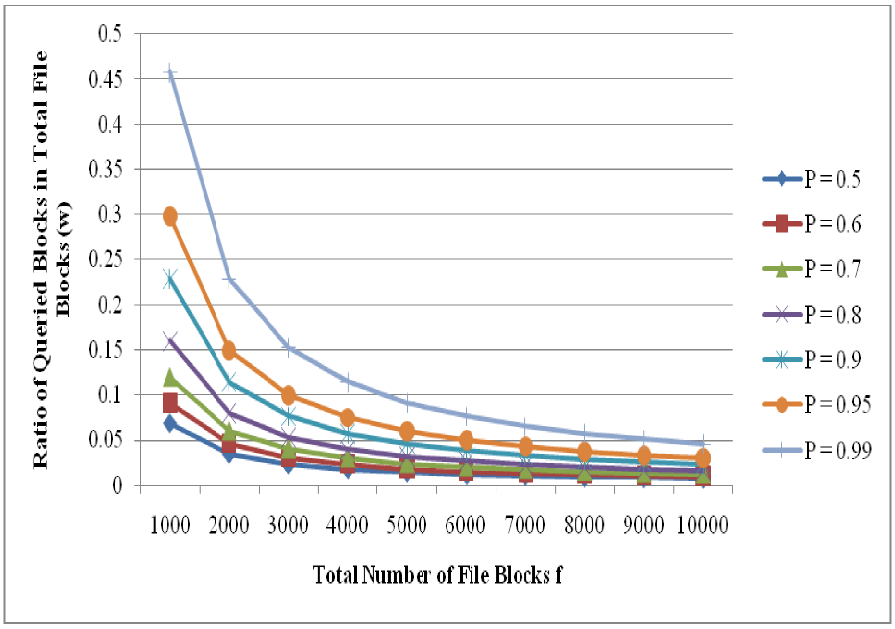

Figure 5: Ratio of Queried Blocks in the Total File Blocks

(ii) Frequency of periodic verification:

Clearly, too frequent audits would lead to a waste of network bandwidth and computing resources for both client and server. On the other hand, fewer number of audits in a given period of time will not be efficient to detect the server misbehaviour during that period. Hence to make this audit scheme more efficient, it is important to choose the optimum audit frequency also. The ratio of queried blocks (w) to the total file blocks under different audit frequency (f) is given by Equation (6).

$$\text{Audit Frequency } f = \frac{\log(1-P_T)}{n.w.T.\log(1-r)} \tag{6}$$

where, P_T is the detection probability during the audit period 'T' , and $r = x / n$

The audit period T can be fixed by the data owner / verifier in advance. Hence the above Equation (6) can be used to analyse the parameter values f and w. From the above equation, it is clear that the audit frequency f is inversely proportional to the ratio of queried blocks w. The analysis also shows that with the increase of audit frequency f, the number of queried blocks decreases at each verification process. Figure 6 shows the relationship between f and w under 10 corrupted blocks (c) for various detection probability (P_X) and audit frequency (f). It is observed that there is a considerable decrease in w with the increase of f. The verifier can choose the appropriate frequency to achieve optimum number of queried blocks.

For example, the verifier asks proof for 766 and 460 blocks for f = 6 and 10 in order to achieve P_X of at least 99%. Hence it is obvious that an appropriate audit frequency would considerably decrease the number of sampled blocks, as well as the computation and communication overhead at both verifier and prover side.

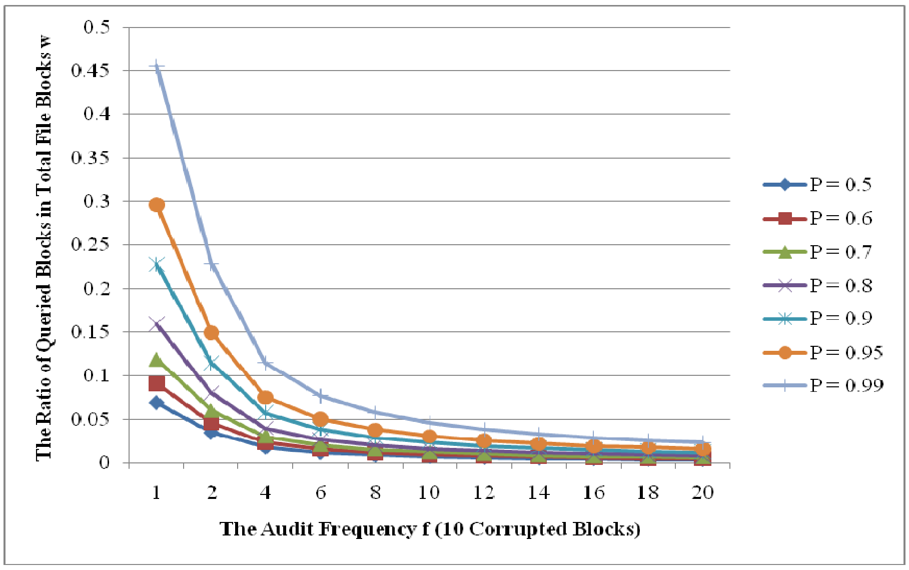

Figure 6: Audit Frequency

5.2 Performance Analysis

From the description of the protocol, it is clear that it involves only minimal communication, computation, and storage costs for both the verifier and the prover.

- **Storage Cost :**

The verifier needs to store only two random numbers, the master key K, the encryption key K_e , and lesser number of tags for sampled group of file blocks. The storage cost incurred by the server includes the encrypted file blocks. No additional meta data needs to be stored by the server.

- **Communication Cost :**

The communication cost is also minimal, since after the encrypted file blocks are outsourced to the Storage Server, the verifier does not need the original file blocks for verification. During Challenge phase also, the verifier transmits only key for i th challenge (k_i) and a random number r2.

- **Computation Cost :**

Let T_{mul}, T_{prf} , T_{prp} , T_{hash} , T_{add} , T_{comp} denote the time cost for multiplication, pseudo-random number generation, pseudo-random permutation, hash operation, addition and comparison operations respectively. In File Setup phase, this protocol needs to generate two random numbers and secret keys, and perform encryption of file blocks. For encryption, this protocol uses symmetric encryption algorithm, which is efficient in terms of computation and storage. This File Setup is a one time process and the theoretical values are difficult to be computed.

In Tag Generation phase, the main computation involves the generation of *m* tags, for which it needs to perform *m* times PRF operation, $m * c$ times PRP operation and hash function. The computation cost for Tag Generation can be given as $mT_{prf} + mT_{add} + mcT_{prp} + 2mcT_{add} + mcT_{hash}$. In Proof Generation phase, this protocol needs to perform *c* times PRP operation and hash function which is given as $cT_{prp} + 2cT_{add} + cT_{hash}$. In Proof Verification phase, it needs to perform only a single comparison operation T_{comp}. The computation cost of various algorithms used in the proposed protocol is summarized and shown in Table 2.

Algorithm	**Computation Complexity**
Tag Generation	$mT_{prf} + mT_{add} + mcT_{prp} + 2mcT_{add} + mcT_{hash}$
Proof Generation	$cT_{prp} + 2cT_{add} + cT_{hash}$
Proof Verification	T_{comp}

Table 2: Performance evaluation of the proposed protocol

5.3 Implementation and Experimental Results

The experiments were conducted in a system with Intel Core i5 CPU with 2.5 GHz speed, 8 GB RAM supporting 64-bit OS. The Operating System used is Windows 10 Home edition. The algorithms were implemented using Java version jdk 1.8.0. Java Cryptography Architecture (JCA) is used for the cryptographic algorithms. For the experiments, files with sizes ranging from 10000 KB to 1 GB are used. The files were split into 8 KB file blocks. All the experimental results are the mean of 10 trials.

Time for File Preparation:

The file preparation time which includes the time for encrypting the original file and then splitting the encrypted file into 8 KB file blocks is almost same irrespective of the confidence levels (99%, 95% and 100%) and linearly increases with the size of the file. The file preparation time for different file sizes upto 1 GB is shown in Figure 7.

Client Computation Time:

The computation time for tag generation at the client side, however, shows the benefit of sampling the data blocks for generating the verifiable tags. When the number of corrupted blocks is equal to 1% of the total number of file blocks, the time required to generate tags at 99% confidence level remains almost constant at an average of 16.02 ms irrespective of the file sizes. At 95% confidence, the tag generation time is 15.50 ms. But, in the absence of sampling, when tags are generated for all the file blocks, the tag generation time linearly increases with file size.

Server Computation Time:

The computation time at server side during proof generation is also constant irrespective of the file size when the challenged blocks are sampled. It is about 18.8 ms at 99% confidence, and 17.45 ms at 95% confidence. But if the entire file is considered for proof generation (100% confidence), then the server computation time increases linearly with file size.

Verification Time:

The verification overhead for detecting 1% corrupted data for various file sizes at 95%, 99% and 100% confidence is compared with the existing works of L. Chen, and E-PDP scheme of G. Ateniese et al. The comparison shows that considering all blocks for verification increases the computation overhead linearly with file size. Spot Checking breaks this linear relationship between the verification time and the file size.

At 99% confidence, the verification overhead of the proposed scheme for any file up to 1 GB in size is about 1.014 ms, while it was 2.24 ms for the scheme of Chen L. and 0.42 s for E-PDP. At 95% confidence, the verification overhead of proposed scheme is 1.009 ms, while it was 1.6 ms for the scheme of Chen L. and 0.4 s for E-PDP. The comparative results of the proposed protocol with the existing schemes are shown in Table 3.

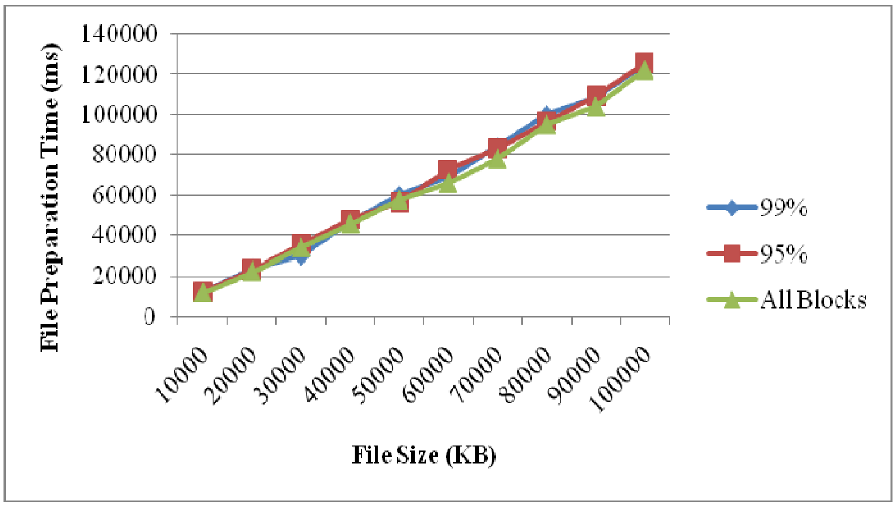

Figure 7: File Preparation Time

Confidence	Proposed Protocol	Protocol by Chen L.	E-PDP
99%	1.014 ms	2.24 ms	0.42 s
95%	1.009 ms	1.6 ms	0.4 s

Table 3: Verification Time of the proposed protocol compared with the existing schemes

6 Conclusion

In order to ensure the correctness of the private and sensitive data of critical sectors, an efficient Remote Data Possession Checking protocol with Random Sampling is proposed which preserves the privacy of user's data, which is an essential requirement mandated by federal laws. The proposed protocol is efficient since it uses spot checking for data integrity verification in which the verifiable tags are computed for a group of randomly sampled data blocks, instead of individual blocks. This reduces the storage as well as computation overhead at the verifier's side as well as prover's side. The security and performance of the proposed protocol is analysed under various parameter values such as file size, number of corrupted blocks, and number of sampled blocks for various confidence levels. It has been shown that the proposed protocol needs only less number of data blocks to detect the server misbehaviour with high probability. The computation overhead is also minimum and remains almost constant for any file of size upto 1 GB when random sampling of file blocks is used. It has been shown that the proposed protocol performs better in terms of verification overhead than the existing E-PDP scheme and the scheme by L.Chen.

Acknowledgements

I sincerely thank Professor C.R.Ranganathan, Adjunct Professor, Coimbatore Institute of Technology, Coimbatore, India for his valuable guidance in the analysis of error detection probability in server misbehaviour detection.

References

[Ateniese et al. 2011] Ateniese G., Burns R., Curtmola R., Herring J., Khan O., Kissner L., Peterson Z., and Song D., "Remote Data Cheking Using Provable Data Possession", ACM Transations on Information an System Seurity, Vol. 14, No. 1, Article 12, 2011.

[Chen 2012] Chen L., “Using Algebraic Signatures to Check Data Possession in Cloud Storage”, Future Generation Computer Systems, 2012

[CSA 2010] "Top Threats to Cloud Computing V1.0", Cloud Security Alliance (CSA), 2010.

[CSCC 2017] "Impact of Cloud Computing on Healthcare Version 2.0", Cloud Standards Customer Council (CSCC), 2017.

[Deswarte et al. 2003] Deswarte Y., Quisquater J.-J., and Saidane A., "Remote Integrity Checking", Proceedings of the IFIP Conference on Integrity and Internal Control in Information Systems (IICIS), pp 1-11, 2003.

[ENISA 2012] "Critical Cloud Computing - A CIIP perspective on cloud computing services, Version 1.0," European Network and Information Security Agency (ENISA), 2012.

[ENISA 2014] "Study on Cryptographic Protocols", European Network and Information Security Agency (ENISA), 2014.

[Erway et al. 2009] Erway C., Kupcu A., Papamanthou C., and Tamassia R., “Dynamic Provable Data Possession,’’ in Proceedings of 16th ACM Conference on Computer and Communication Security (CCS), 2009, pp. 213-222.

[EU GDPR 2016] "Regulation (EU) 2016/679 of the European Parliment and of the Council", EU General Data Protection Regulation, 2016.

[Feng and Long 2017] Feng J. and Long S., "Data Integrity Checking Protocol with Data Dynamics in Cloud Computing", International Journal of Communications, Network and System Sciences, 10, pp. 274-282, 2017.

[Hao et al. 2011] Hao Z, Zhong S, Yu N, "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability". IEEE Transactions on Knowledge and Data Engineering, 23(9), 1432–1437, 2011.

[Jules and Kaliski 2007] Jules.A. and Kaliski B.S.,"PORs: Proofs of retrievability for large files", Proceedings of the 14th ACM conference on Computer and Communications Security (CCS'07), ACM, New York, 2007.

[More and Chaudhari 2016] More S. and Chaudhari S., "Third Party Public Auditing scheme for Cloud Storage", Procedia Computer Science, Elsevier, 79(2016), 69-76, 2016.

[Saxena and Dey 2016] Saxena R. and Dey S., "Cloud Audit: A Data Integrity Verification Approach for Cloud Computing", Procedia Computer Science, Elsevier, 89(2016), 142-151, 2016.

[Sebe et al. 2008] Sebe F., Domingo-Ferrer J., Martinez-Balleste A., Deswarte Y., and Quisquater J-J., "Efficient Remote Data Possession Checking in Critical Information Infrastructures", IEEE Transactions on Knowledge and Data Engineering, VOL 20, NO. 8, 2008.

[Singh and Pasupuleti 2016] Singh A. P. and Pasupuleti S. K., "Optimized Public Auditing and Data Dynamics for Data Storage Security in Cloud Computing", Procedia Computer Science, Elsevier, 93 (2016), 751 – 759, 2016.

[Wang C. et al. 2011] Wang C, Sherman S.M.Chow, Wang Q, Ren K, and Lou W, “Privacy-Preserving Public Auditing for Secure Cloud Storage”, IEEE Transactions on Computers, 2011.

[Wang Q. et al. 2011] Wang Q, Wang C, Ren K, Lou W, and Li J, “Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing”, IEEE Transactions on Parallel and Distributed Systems, Vol 22, No. 5, pp. 847-858, 2011.

[Wang et al. 2012] Wang C, Wang Q, Ren K, Cao N, and Lou W, “Toward Secure and Dependable Storage Services in Cloud Computing”, IEEE Transactions on Services Computing, Vol. 5, No. 2pp. 220-232, April-June 2012.

[Wei et al. 2017] Wei J., Niu X., Zhang R., Liu J. and Yao Y., "Efficient data possession–checking protocol with deduplication in cloud", International Journal of Distributed Sensor Networks, Vol. 13(8), p.p. 1-13, 2017.

[Yu et al. 2014] Yu Y., Au M., Mu Y., Tang S., Ren J, Susilo W and Dong L, "Enhanced privacy of a remote data integrity checking protocol for secure cloud storage", International Journal of Information Security, online first 1-11, 2014.

[Zhu et al. 2010] Zhu Y., Wang H., Hu Z., Ahn G.-J., Hu H., and Yau S. S., “Cooperative provable data possession.” Cryptology ePrint Archive, Report 2010/234, 2010. http://eprint.iacr.org/.