On-demand Integrity Verification Technique for Cloud Data Storage

Rajat Saxena
and
Somnath Dey
Cloud Computing Lab
Department of Computer Science and Engineering
Indian Institute of Technology Indore, Indore, India

Data Integrity Verification (DIV) is one of the primary security check for cloud data. Cloud users have assured the safety of their data with frequent checking of data integrity. In this paper, we provide an efficient DIV approach based on a privacy-preserving audit structure. The main building components of our approach is a multi-power variant of the Paillier cryptography system with homomorphic tag. Paillier cryptography system along with homomorphic tag assigns a unique and verifiable value to each data block that helps to perform dynamic data operations in cloud environment. To demonstrate our approach, we have implemented an application on Hadoop and MapReduce framework. Experimental results establish the efficiency of the proposed scheme which outperforms existing schemes between 8-12% on various parameters.

Keywords: Homomorphic Tag, Paillier Homomorphic Cryptography(PHC), Proof of Retrievability (PoR), Provable Data Possession (PDP), Third Party Auditing.

1. INTRODUCTION

Cloud computing is the innovative on-demand automation of the services to process, organize and store on the remote Cloud servers. Cloud environment suggests virtualization of resources to fulfill the service requirements of the Cloud users. The Internet Protocol (IP) and other networking protocols have used to access the resources. Data integrity verification (Saxena and Dey, 2016, 2014) is one of the massive responsibility with cloud data, because the probability of involvement in the malicious activity of a cloud users is very high. There are many ways to address this problem. The user can use encryption and decryption process. However, it requires huge computing time and functional overheads. Data auditing methods are the other way to address this problem. There are three types of data auditing techniques.

1. Provable Data Possession (PDP):. PDP (Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song, 2007) requires two steps : First, the client (verifier) preprocesses the data and keeps a small amount of metadata. Then, client sends whole data to an untrusted data storage server (prover) for storing. In the next step, client (verifier) verifies with the help of metadata that the data stored in the storage server still possesses the client's original data and stored data has not been tampered or deleted. However, the PDP technique has the following limitations.
—It works only for the static data.
—It applies only for encrypt files that allow a limited number of queries.

2. Proof of Retrievability (PoR):. It (Juels and Jr., 2007) has two steps : First, the client (verifier) stores a file on an untrusted data storage server or prover. In the next step, the client runs data audit proof algorithm. This proof helps prover to ensure that it still possesses the client's data file and client can recover the entire data. However, the PoR technique has the following limitations.

—The effectiveness of these schemes primarily rests on the preprocessing steps that the user conducts before out-source the data file. It introduces significant computational and communication overheads.
—It introduce tradeoff between storage overhead and cost of communication, thus some of this techniques store less storage with high cost.

*3. **Third Party Auditing**.* It (Wang, Wang, Ren, and Lou, 2010) assigns auditing work to single Third Party Auditor (TPA). The TPA should be able to efficiently audit the cloud data storage without demanding the local copy of data and introducing no additional on-line burden to the cloud user. However, the TPA technique has the following limitations.
—All the above schemes provide only binary results about the storage status for identifying misbehaving server(s).
—Single TPA cannot handle SLA and legal issues for data possession and prone to single-point failure.

To address the above problems, we propose a on-demand data verification technique in which client handle various audit sessions from different users for their outsourced data files on Cloud Service Provider(CSP). We assume for our system that Cloud Service Provider (CSP) must be trustworthy.

In this paper, we propose an efficient and privacy preserve audit structure with two-tier architecture. Major contributions of our approach are as follows:

(1) We provide on-demand public audit structure for privacy preservation and data storage security in the cloud environment.
(2) The proposed method also supports dynamic data operations using the Paillier cryptography system.
(3) Proposed scheme can also handle various efficient and synchronous audit sessions from multiple users.

Organization. We have organized the remaining paper into following sections. In section 2, we have discussed the related work for data integrity verification. Section 3 presents our proposed scheme for data integrity verification. In section 4, we describe the support of dynamic data operations. Security and performance analysis are presented in Section 5. Finally, section 6, concludes our work.

2. RELATED WORK

The initial PDP technique (Ateniese et al., 2007) has used Homomorphic Verifiable Tags (HVT) as the key tool for DIV. Other approach for PDP is scalable PDP technique (Ateniese, Pietro, Mancini, and Tsudik, 2008) that used the symmetric key cryptography for DIV. A variant of this approach is dynamic PDP technique (Erway, Küpçü, Papamanthou, and Tamassia, 2009). It has used rank-based authenticated dictionary, skip list, and RSA tree for DIV. All the above approaches are probabilistic and supports sampling with the $[1-(1-p)^c]$ probability of server misbehavior detection, where c is challenged blocks.

In efficient PDP technique (Sebé, Domingo-Ferrer, Martínez-Ballesté, Deswarte, and Quisquater, 2008), the asymmetric key cryptography (RSA modules) is used as key tool for DIV. Another PDP technique (Chen, 2013) uses the algebraic signature for DIV. Both techniques are probabilistic approach that supports sampling with the $[1-(1-p)^{c*s}]$ probability of server misbehavior detection, where c is challenged blocks and s is the numbers of sectors in blocks.

PoR (Juels and Jr., 2007) technique has used the error correcting codes, symmetric key cryptography, sentinel creation, and permutation for DIV. Other work PoR-HA (Dodis, Vadhan, and Wichs, 2009) has used the error correcting codes, Reed-Solomon codes, and hitting sampler for DIV. In PoR-TI (Bowers, Juels, and Oprea, 2009), the adversarial error correcting codes is used for DIV. All the above work are probabilistic that supports sampling with the $[1-(1-p)^c]$ probability of server misbehavior detection, where c is challenged blocks.

In C-PoR (Shacham and Waters, 2008), the BLS signature and pseudo-random functions are used for DIV. It is a probabilistic approach that supports sampling with the $[1-(1-p)^{c*s}]$ probability of server misbehavior detection, where c is challenged blocks and s is the numbers of sectors in blocks.

In HAIL (Juels, Bowers, and Oprea, 2009), integrity protected error correcting universal Hash and MAC function are used for DIV. It may be probabilistic or deterministic approach that supports sampling with the $[1-(1-p)^{c*s}]$ probability of server misbehavior detection, where c is challenged blocks and s is the numbers of sectors in blocks.

However, there are many difficulties with PDP and PoR techniques. These techniques can be employed only for encrypted files and allow only a limited number of queries. Further, the above schemes are not suitable for the batch auditing in the cloud environment because of their computation overhead. The benefits of these techniques only depend on the preprocessing steps which user can apply on outsource data file. There is a settlement between dynamic data operations and privacy preservation. But some of the schemes do not preserve privacy. There is a trade-off between the cost of communication and storage overhead. However, some of these PDP schemes require a high cost for less storage.

There are some schemes (Wang et al., 2010; Wang, Wang, Li, Ren, and Lou, 2009; Hao, Zhong, and Yu, 2011; Zhu, Hu, Ahn, and Yu, 2012) which assign audit tasks to single Third Party Auditor (TPA). TPA is an authentic and trusted entity that independently manages the data audits. The above schemes are not suitable for the batch auditing in the cloud environment because of their computation and third-party overhead.

In (Wang et al., 2010) technique, bilinear map, MAC, and homomorphic authenticator are used for DIV. It may be probabilistic or deterministic approach that supports sampling with the $[1-(1-p)^c]$ probability of server misbehavior detection, where c is challenged blocks.

In (Wang et al., 2009) technique, merkle hash tree and aggregate signature are used for DIV. It is a probabilistic approach that supports sampling with the $[1-(1-p)^{c*s}]$ probability of server misbehavior detection, where c is challenged blocks and s is the numbers of sectors in blocks.

In Hao et. al (Hao et al., 2011), RSA based bilinear homomorphic verifiable tags are used for DIV. It is a deterministic approach that supports sampling with the $[1-(1-p)^{c*s}]$ probability of server misbehavior detection, where c is challenged blocks and s is the numbers of sectors in blocks.

In cooperative PDP (Zhu et al., 2012), homomorphic verifiable hash index hierarchy is used for DIV. It is a probabilistic approach that supports sampling with the $[1-\prod p_k \varepsilon p(1-p_k)^{r_k*c*s}]$ probability of server misbehavior detection, where n is the block number, c is the sampling block number and s is the numbers of sectors in blocks. p is the probability of block corruption in a cloud server and P_k is the probability of k^{th} cloud server in a multi-cloud.

In these schemes, single TPA cannot handle SLA and legal issues for data possession and prone to single-point failure. For these schemes, error localization is very difficult to find. All the above schemes provide only binary results about the storage status for identifying misbehaving server(s). None of this scheme supports multiple TPAs for cross checks and cross authenticate the data integrity verification, privacy preservation and computation accuracy. There is a tradeoff between data dynamics, privacy preservation and public verifiability in these schemes. TPA may simultaneously handle various audit sessions from different users for their outsourced data files by multi-user setting during efficient auditing process.

Therefore, Saxena et al. (Saxena and Dey, 2016, 2014, 2017) have proposed multiple TPA schemes, in which various synchronous audit sessions from different users are handled by each single TPA simultaneously. But, it has computational overheads.

To address the above limitations, we have proposed an on-demand data verification approach, which describes in next section with more details.

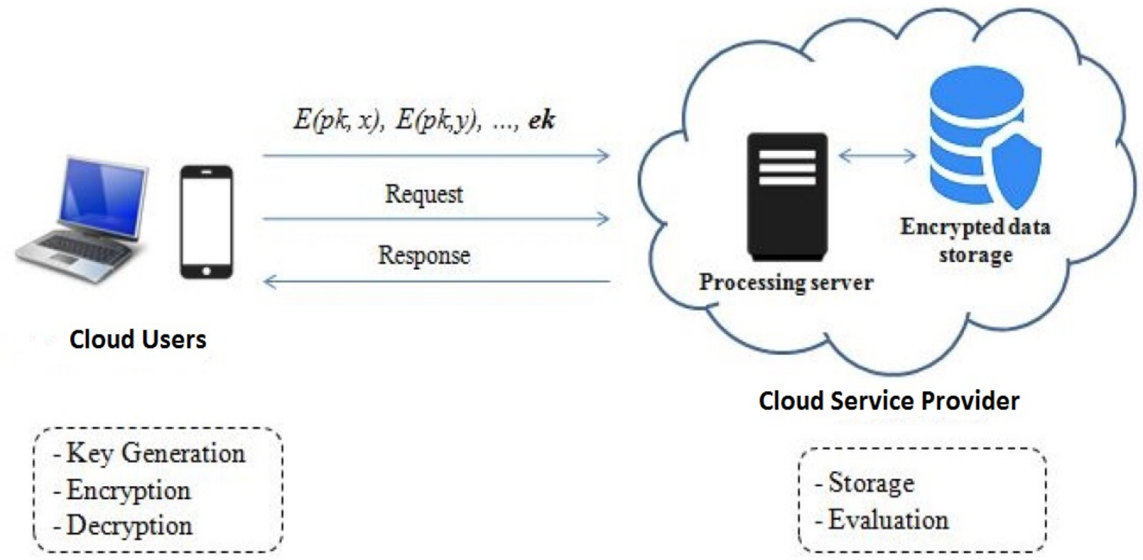

Figure 1: Our Approach for Privacy Auditing

3. PROPOSED SCHEME

In this section, first, we present system models of our scheme.

3.1 Proposed Data Integrity Verification Scheme

In this paper, we propose a two-tier architecture for privacy preserve auditing. The first tier is cloud user and the second tier is cloud service provider which has two parts (1) Processing Server, and (2) Encrypted data storage. The details of each tier have given as follows:

(1) **Cloud Users**
Any consumer of cloud service can serve as a cloud user. Our assumption for cloud user is that they have limited resources and are not capable of performing computation-intensive jobs, such as different auditing tasks (privacy preserving and data integrity verification).

(2) **Cloud Service Provider(CSP)**
CSPs are the organizations who provide the sufficient infrastructure and resources to as a service to the end users. Distribution of resources could be on many servers which are situated at different places. CSP has two parts:
(1) Processing Server: This is used to compute and process the user's request.
(2) Encrypted Data Storage: This is used to store encrypted data and evaluation key for the convenience of cloud users.

3.2 Work Flow of the Proposed Method

In this section, we present the process flow of our proposed audit scheme. In first step, a cloud user generates keys for encryption and decryption of data using key generation process. Next, the user encrypts data with Multi-power RSA algorithm using encryption process and sends to CSP to store the encrypted data. CSP stores the encrypted data in its encrypted data storage. When a user requests processing server to evaluate the data, the CSP will perform the evaluation process and return the encrypted to cloud user. Finally, cloud user decrypts the encrypted data. The workflow of the two-tier system architecture is described in Figure 1. The detail description of these tasks is given below.

(1) **Key Generation**
In this step, the cloud user generates two keys, the evaluation key (e_k) and the private key (p_k) at the client side. For this purpose, we use variant of Paillier Cryptography System (Paillier, 1999). Details of Paillier Cryptography System is given in Appendix 6.2. The key generation algorithm takes two inputs n and r as parameters. We generate two distinct primes p and q in which each number is [n/r+1] bits long and used to generate private and evaluation key. We have chosen an integer e which is relatively prime to L and p. This integer is used to generate private key. The process for generation of a key pair as follows:

Algorithm 1 : Key Generation

```
Input: $n, r$ .
Output: The evaluation key is $e_k$, The Private key is $p_k$.
1: Compute $N = p^r q$ and $L$ =(p - 1)(q - 1).
2: Compute $d \equiv (e^{-1}) mod\ L$.
3: Compute $d_p \equiv (d) mod\ p$ and $d_q \equiv (d) mod\ q$.
4: Compute $k_p = \frac{N}{P_r}(\frac{N}{P_r})^{-1}$; where $k_p \equiv (1) mod\ p^r$ and $k_q \equiv (1) mod\ q$.
5: Output: The evaluation key is $e_k = (N)$; the private key is $p_k = (N; p; q; r; e; d_p; d_q; k_p; k_q)$.
```

(2) **Encryption**
The Cloud user divides the file F into a number of file blocks $m_1; m_2; ....; m_k$. The size of data blocks is $512MB$. The cloud user encrypts these blocks with its private key (p_k)using RSA technique (Boneh, Durfee, and Howgrave-Graham, 1999) and Euler's totient function. After that user sends the encrypted data and the evaluation key (e_k) to the CSP for storage. Given the private key p_k, let $m_i \in \mathbb{Z}_N$ be a plaintext. The ciphertext c_i is computed as follows:

$$c_i \equiv (m_i^e) mod\ N \tag{1}$$

(3) **Storage**
At the CSP server, file blocks are presented in encrypted form as the individual ciphertexts $(c_1; c_2; ::: c_k)$. These encrypted blocks and the evaluation key (e_k) is stored on the cloud data storage server for the verification purpose.

(4) **Request**

The cloud user sends a verification request to the Cloud server. The CSP server searches and recomputes the individual block information from the ciphertexts of each block using evaluation process.

(5) **Evaluation**
In this process, the CSP server recomputes overall ciphertext from the ciphertext of each block. To do this, it takes the cross product of the ciphertext of each block. If C is overall ciphertext and $(c_1; c_2; ::: c_k)$ are the ciphertext of each block, then C is generated using equation 2.

$$C = Eval(e_k; c_1; c_2; ....; c_k) \tag{2}$$

The processing server needs to retrieve and verify the encrypted data $(c_1; c_2; ::: c_k)$. To do this, CSP will provide overall individual c_i information given in equation 2. Now, ciphertext of each block is computed using m_i^e. It recomputes the cross product of individual block information $(m_1 \times m_2 \times ::: \times m_k)^e\ mod\ N$ by using the evaluation key e_k. Now, CSP will able to evaluate the whole cipertext C.

$$\begin{aligned}
&\equiv \quad (m_1^e \times m_2^e \times ..... \times m_k^e) mod\ N \\
&\equiv \quad (m_1 \times m_2 \times ::: \times m_k)^e mod\ N. \\
&\equiv \quad E(p_k, m_1 \times m_2 \times ..... \times m_k). \\
&\equiv \quad C.
\end{aligned}$$

(6) **Response**
The cloud service provider returns the processed result to the cloud user that is in encrypted form. With the help of decryption method, the user can evaluate and verify the plaintext M.

(7) **Decryption**
For the given private key p_k, user utilizes the Hensel lifting to decrypt the ciphertext C and construct a plaintext. For this operation, modulus of (M_p^e) has conducted with respect to p^r. It has done for all values from 1 to r-1. After this, the user performs the Chinese remaindering to recover the plaintext. To do this, user calculates $(M_p) mod\ p^r$ and $(M_q) mod\ q$ that alternatively generates M. The details of the Hensel lifting theorem and Chinese remaindering theorem have described in Appendix 6.1.

Algorithm 2 : Decryption

Input: a private key $p_k = (N; p; q; r; e; d_p; d_q; k_p; k_q)$ and a ciphertext $C \in \mathbb{Z}_N$.
Output: $M \in \mathbb{Z}_N$.

```
1: Compute $C_0 \equiv (C) mod\ p$; $C_p \equiv (C) mod\ p^r$ and $C_q \equiv (C) mod\ q$.
2: Compute $M_0 \equiv (C_0^{d_p})\ mod\ p$ and $M_q \equiv (C_q^{d_q})\ mod\ q$.
3: Using the Hensel lifting, we construct a plaintext $M_p\ modulo\ p^r$ such that $C_p \equiv (M_p^e) mod\ p^r$.
   To do this, first we set $K_0 \leftarrow M_0$.
4: **For** i=1 **to** r-1 **do**
5: $E_i \leftarrow (K_{i-1}^e) mod\ p^{i+1}$.
6: $F_i \leftarrow (C_p - E_i) mod\ p^{i+1}$.
7: $G_i \leftarrow F_i / p^i$.
8: $M_i \leftarrow G_i \times ((eM_0^{e-1})^{-1}) mod\ p$.
9: $K_i \leftarrow K_{i-1} + p^i M^i$.
10: **end for**
11: Using the Chinese remaindering, recover the plaintext $M = m_1 \times m_2 \times .... \times m_k$ such that
    $M \equiv (M_p) mod\ p^r$ and $M \equiv (M_q) mod\ q$. To do this, first set $M_p \leftarrow K_{r-1}$.
12: Compute $M \leftarrow (M_p \times k_p) + (M_q \times k_q) mod\ N$.
13: Output: $M \in \mathbb{Z}_N$.
```

3.3 Correctness Proof of Effective Privacy Preservation

In this part, we shell prove the correctness of our approach for the effective privacy preservation. Let $N = p^r q$ where p; q are two distinct primes and $r \geqslant 2$. Let e; d be two integers satisfying $ed \equiv (1)\ mod (p-1)(q-1)$ and gcd(e; p) = 1.

We define the threat vector in this part. Here, we select $r \geqslant 2$ for making the factorization difficult. For this value of r, any crypto-logical system will fail to decrypt the information from the stored data.

We denote the ciphertext reduced modulo q by C_q, the ciphertext reduced modulo p by C_0 and the ciphertext reduced modulo p^r by C_p.

Suppose that $d_p \equiv (d) mod\ p-1$ and $d_q \equiv (d) mod\ q-1$ and i.e., $\exists\ a_1; a_2 \in \mathbb{Z}$ such that $d = d_p + a_1\ (p-1)$ and $d = d_q + a_2\ (q-1)$.

According to the equation 1, we have the ciphertext $C \equiv (M^e) mod\ N$ where $M \in \mathbb{Z}_N$. Let us decrypt C modulo p

$M_0 \equiv (C^d) mod\ p$

$\equiv (C_0 + bp^d) mod\ p$; where $b \in \mathbb{Z}$.

$\equiv (C_0^{d_p + a_1(p-1)}) mod\ p$

$\equiv (C_0^{d_p}) mod\ p$ Thanks to Theorem 6.3

Similarly, we have, $M_q \equiv (C_q^{dq}) mod\ q$

The relationship between the ciphertext $C_p$and the plaintext M_p is $C_p \equiv M_p^e (mod\ p^r)$. Suppose that M_p has p-adic expansion such that $M_p \equiv M_0 + pM_1 + p^2 M_2 + ... + p^{r-1} M_{r-1} mod\ p^r$, where $M_0; ....; M_{r-1} \in \mathbb{Z}_p$.

Let the function $A_i(M_0; M_1; ....; M_i)$ be defined as follows:

$C_p \equiv A_i(M_0; M_1; ....; M_i) \equiv (M_0 + pM_1 + p^2 M_2 + ... + p^i M_i)^e$; where $1 \leqslant i \leqslant r - 1$.

Let us reduce C_p *modulo* p^{i+1} by using the binomial theorem 6.2 ,

$A_i(M_0; M_1; ....; M_i) \equiv (M_0 + pM_1 + p^2 M_2 + ... + p^i M_i)^e mod\ p^{i+1}$

$\equiv \sum_{k=0}^{e} \binom{e}{k} (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^(e - k)(p^i M_i)^k mod\ p^{i+1}$

We can explore this equation with following expansion:

$\equiv (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^e + (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^{(e-1)} ep^i M_i + \sum_{k=2}^{e} \binom{e}{k} (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^(e - k)(p^i M_i)^k mod\ p^{i+1}$

It can be divided into two subparts as follows:

$\equiv (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^e + ep^i M_i \times \sum_{j=0}^{e-1} \binom{e-1}{j} M_0^{e-1-j} \times (pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^j mod\ p^{i+1}$

$\equiv (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^e + p^i e M_0^{e-1} M_i mod\ p^{i+1}$

We note that $A_{i-1} \equiv (M_0 + pM_1 + p^2 M_2 + ... + p^{i-1} M_{i-1})^e$ for $i = 0; 1; .....; r - 1$.

Thus, $C_p = A_{i-1} + p^i e M_0^{e-1} M_i mod\ p^{i+1}$

To follow this, the values $M_0; M_1; ....; M_{r-1}$ can be recursively calculated by the relationship

$$M_i \equiv (\frac{C_p - A_{i-1}(mod\ p^{i+1})}{p_i}) \times e^{-1} M_0^{1-e} mod\ p \qquad (3)$$

It is observable that since e is relatively prime to p and $M_0 \in \mathbb{Z}_p$, so e and M_0^{e-1} have reverse modulo p. It terminates the correctness proof for effective privacy preservation.

4. SUPPORT FOR DYNAMIC DATA OPERATIONS

Our approach supports three dynamic data operations, which are described in the following sub-sections.

4.1 Update Operation

In some situations, the user modifies the existing file blocks, and this modification restructures the whole file. This operation is known as the update operation. To implement the update operation, we use the first homomorphic property of a Paillier cryptography system. This property is described in our notation as follows:

Let m_1, m_2 be any two file blocks containing plain-texts such that m_1, $m_2 \in \mathbb{Z}_J$ and r_1, r_2 be two random numbers such that r_1, $r_2 \in \mathbb{Z}_J^*$. First homomorphic property says that the decryption of the product of two cipher-texts is equal to the sum of their corresponding plain-texts. This could be represented by the following equation.

$$D[E(m_1,r_1).E(m_2,\ r_2)] = D[E(m_1+m_2,\ r_1.r_2) \mod J^2] = m_1+m_2 \mod J \qquad (4)$$

Thus, we can get the dynamic update operation and the addition of two plain-text without retrieving the plain-texts.

To update a file block m_i, user modifies this block with small change $\pm\Delta m_i$. It causes new block $m_i \pm \Delta m_i$. Thus, change in ciphertext will be $(m_i \pm \Delta m_i)^e$. It results the final ciphertext as $c_i \pm \Delta c_i$, after the commit operation.

4.2 Append Operation

Sometime, the user may add new data block at the end of the stored files which causes an increment in the size of the stored data. This operation has referred as the append operation. To implement this operation, we use the second homomorphic property of a Paillier cryptography system, which is described in our notation as follows :

For any $m_1, m_2 \in \mathbb{Z}_J$ and $r_1, r_2 \in \mathbb{Z}_J^*$, decryption of an encrypted plain-text raised to the power of another plain-text is equal to the product of two plain-texts. The following equations can be express it.

$$D[E^{m_2}(m_1,r_1)] = D[E(m_1.m_2,\ r_1^{m_2}) \mod J^2] = m_1.m_2 \mod J \qquad (5)$$

$$D[E^{m_1}(m_2,r_2)] = D[E(m_1.m_2,\ r_2^{m_1}) \mod J^2] = m_1.m_2 \mod J \qquad (6)$$

Thus, we could ensure the dynamic appending of data and multiplication of two plain-text without retrieving plain-texts.

4.3 Delete Operation

In some occasions, after saving the data in the cloud, the user needs to delete some data blocks. We consider this operation as the delete operation. It can be considered as a specialized update operation because we can replace the original data blocks with null blocks or any special predetermined symbol blocks.

There are two types of delete operation: (1) Partial block delete operation, in which CSP retrieves the file from the storage server in encrypted form and some part of blocks is deleted, and modified blocks are stored on the server. (2) Full block delete operation, in which CSP retrieves the file from the storage server in encrypted form and the full blocks of file are replaced with null blocks. These null blocks are assigned to the scheduler for the fresh allocation.

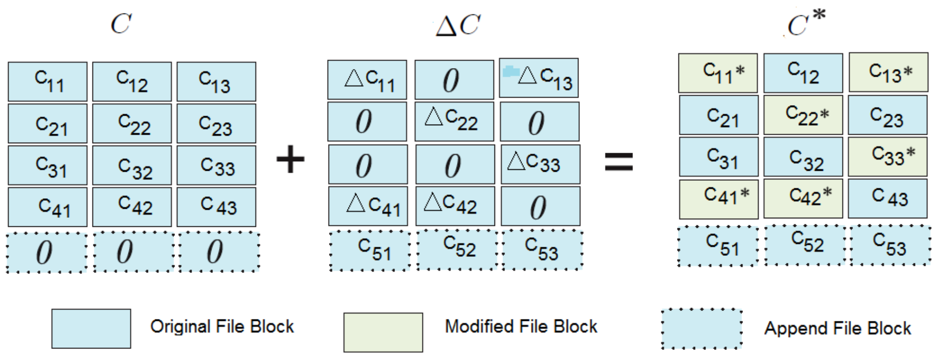

Figure 2: Explanation of Update and Append Operation

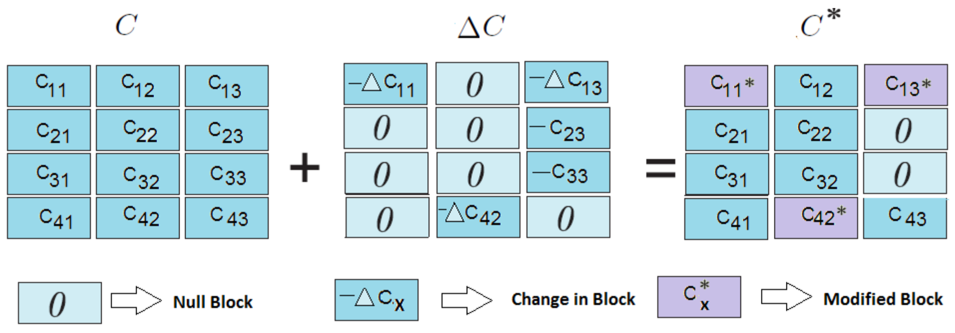

Figure 3: Explanation of Delete Operation

4.4 Use Case

We have taken a Use Case to show the different dynamic operations performed by a user in the data file.

(1) **Update Operation:**
Cloud user requests to CSP for the update operation in a file. CSP retrieves this file from the storage server in encrypted form and uses the first homomorphic property of a Paillier cryptography system as outlined in equation 4. This property enables server to update the file. Pictorial representation of update operation is given in Figure 2. In Fig. 2, we represent data blocks of a file as the C matrix and the actual file block as $c_{13}, c_{33}, c_{41},$ *and* c_{42}. If the modification or updation in file block have described as $-\triangle c_{13}, \triangle c_{33}, \triangle c_{41}, and\ \triangle c_{42}$, then $c^*_{13}, c^*_{33}, c^*_{41},$ *and* c^*_{42} are depicted as updated output blocks.

(2) **Append Operation:**
Cloud user requests to CSP for the append operation in the file. CSP retrieves this file from the storage server in encrypted form and uses the second homomorphic property of a Paillier cryptography system as outlined in equation 5 and 6. This property enables the server to append the data to the file. Pictorial representation of append operation is given in Figure 2. In Figure 2, we represent data blocks of a file as the C matrix and and the appended file block as $c_{51}, c_{52},$ *and* c_{53}. After the commit operation, these blocks are added in file matrix C^*. Description of append operation is given in Figure 2.

(3) **Delete Operation:**
Cloud user requests to CSP for the delete operation in a file. In partial delete operation, only

some partial block information is deleted, and blocks are converted into the modified blocks. The modified blocks coexist after the commit operation on blocks. We represent data blocks of a file as the C matrix. The explanation of Delete operation is shown in Figure 3.

In Figure 3, if three blocks such as $c_{11}, c_{13},\ and\ c_{42}$ are selected for the partial delete operation, and $-\triangle c_{11}, -\triangle c_{13},\ and\ -\triangle c_{42}$ are are in information to be deleted from the respective blocks. Then, the modified blocks are depicted as the block $c^*_{11}, c^*_{13},\ and\ c^*_{42}$.

In full block delete operation, entire blocks are to be deleted. After deletion of full blocks, blocks are replaced with the null blocks. After the commit operation, these null blocks are allocated as a fresh data blocks into the scheduler, which can be assigned for storing the information.

In Figure 3, if two blocks such as $c_{23}\ and\ c_{33}$ have selected for the full delete operation, and $-c_{23}\ and\ -c_{33}$ are the block change, then the modified blocks are represented by null blocks as shown in Fig. 3 which can be assigned to the scheduler as a fresh data blocks.

5. ANALYSIS

This section, first, presents the security analysis of our proposed method using the binding property of the Paillier cryptography system. Further, we have discussed the experimental results to judge the performance of our scheme.

5.1 Security Analysis

The security of our scheme depends on the difficulty of factoring the modulus N formed of two or more distinct prime numbers as well as of finding the decryption exponents, where the encryption exponent is also private. The fastest factorization algorithms, e.g., the Number Field Sieve and the Elliptic Curve Method (Lenstra Jr, 1987), cannot take advantage of the modulus $N = p^r q$ structure, if it gives the primes factorization of the modulus N. To expose the decryption exponent d an adversary must find two exponents: e and d such that $ed \equiv 1\ (mod\ (p-1)(q-1))$ and the decrypted data are semantically correct.

The goal of our technique is to ensure that solely authorized users can read, use, or contribute to the data stored at CSP. These security controls add another layer of stability against possible threats by cloud users, administrators and other vulnerable actors on the network. To ensure better security, we use the self-binding property of Paillier cryptography system, which is described in our notation as follows:

$$[E(m_1, r_1).r_2^J] \mod J^2 = E(m_1,\ r_1.r_2) \tag{7}$$

With this property, any cipher-text can change to another cipher-text without affecting the plain-text. We utilize this property to make the job of adversary very difficult to predict the plain-text.

5.2 Performance Analysis

We have executed our experiments with the setup given in Table I. We have configured a cloud environment to perform audit task of the stored files of the cloud users.

Table I: Experimental Setup

No of PCs	2
Processor	Intel core i7-2600S 2.80 GHz
Memory	16 GB
File Storage Server	Citrix Xen Server 6.2.0 (Xenserver, 2014)
Cloud Environment	Cloudera CDH 5.3.0-0 (Cloudera, 2014)
Maximum Data for Storage	1TB

For the implementation, we have followed the recommendations for generating a key pair in (Boneh et al., 1999; Wiener, 1990). We have set three modulus N_1, N_2 and N_3, each 1024 bits long formed of distinct primes as mentioned in the Table II. Then, we have picked an encryption exponent $e = 216 + 1$. Implementation is provided the following Table II, that represents the required time to decrypt a quantity of data (from 100 to 1000 bits long) under our scheme and two possible variants. We denote the traditional RSA modulus by N_1 and our RSA modulus by N_3. N_2 represents a modulus formed from three distinct primes.

Table II: Decryption Time Performance

Data size (bits)	$N_1 = p.q$	$N_2 = p_1.p_2.p_3$	$N_3 = p^2.q$
100	44.25 ms	24.5 ms	21 ms
200	48.75 ms	25.5 ms	20.33 ms
300	52.66 ms	25 ms	20.75 ms
400	56.33 ms	24.75 ms	21.66 ms
500	58.75 ms	26 ms	20.8 ms
600	59.25 ms	25.2 ms	21.25 ms
700	61.8 ms	24.66 ms	20.66 ms
800	66.25 ms	25 ms	20.75 ms
900	71.4 ms	27 ms	21.8 ms
1000	78.33 ms	26.66 ms	21.33 ms

The results of decryption time performance show that our RSA scheme gets a decryption speedup by the factor of 2.84 on average over the traditional RSA scheme. Whereas, when using the RSA modulus formed from three distinct primes and employing the Chinese remaindering to decrypt, cloud user gets a decryption speedup by the factor of 2.35 on average over the traditional RSA scheme. It seems certain that our RSA scheme gives a significant speedup. The running time of our RSA decryption is approximately stable and thanks to the relationships of the Hensel lifting and the Chinese remaindering steps.

The best option to analyze the performance of any method is observing the performance parameters. We have measured the experimental results with respect to four performance parameters and compare with other methods. The observation of different parameters are given in the following:

(1) **Detection Probability of Server Misbehavior:** In our scheme, we can set a number of queried data blocks and a number of challenged data blocks according to user's requirement. When users wish to store data for a short period, users can use less amount of challenged blocks to reduce the overload of CSP. We compute P_X, the probability that at least one of the blocks picked by CSP matches one of the blocks deleted by the server, with the following equation.

$$P_X = P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \{\frac{n-r}{n}.\frac{n-1-r}{n-1}.\frac{n-2-r}{n-2}....\frac{n-c+1-r}{n-c+1}\} \quad (8)$$

P_X indicates the detection probability of server misbehavior that depends on a total number of file blocks n, deleted blocks r and challenged blocks c. If storage server deletes r blocks of the file, then the CSP will detect server misbehavior after a challenge for c blocks.

For 1 TB file size, default HDFC file block size = 64 MB, n = 16384 blocks, r = 164 blocks, c = 460 blocks, total number of sample = 1000, audit frequency (Max 95Hz), and number of users =1000, we find comparative results with other methods which is shown in Figure 4.

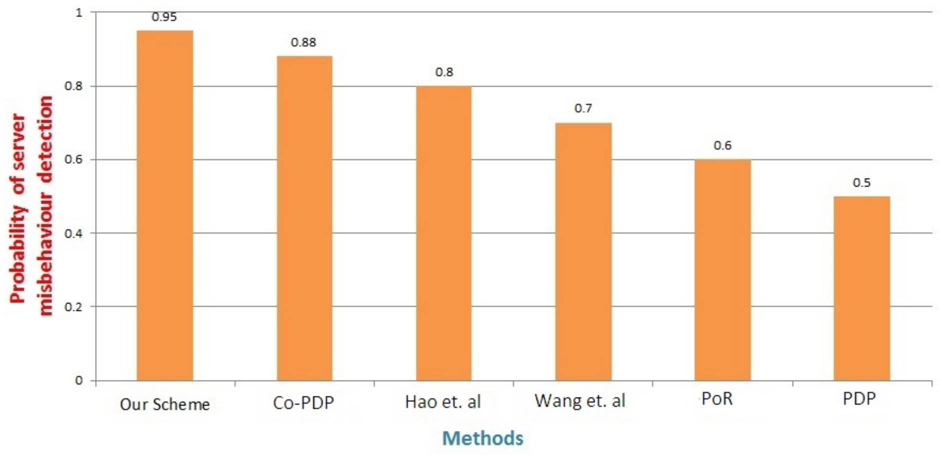

Figure 4: Detection Probability of Server Misbehaviour for Our Method and Other Techniques

(2) **Verification Delay:** Verification delay is the time difference between the time of verification completion and the time of client request for verification. It is a major factor for the verification of any data

$$Verification\,Delay\ =\ Time\,at\,verification\,complete{-}Time\,of\,client\,request\,for\,verification \tag{9}$$

We have computed the verification delay with the same configuration as described earlier and find comparative less verification delay which is shown in Figure 5. The time complexity of our experiments to calculate verification delay is $O(c*k)$, where c is the number of required blocks for verification and k is the number of users.

Table III describes the parameter assessment of our approach. It demonstrates that when the total number of sample increases, the sensitivity (Detection Rate) and accuracy also increase. The reason behind this is the increment in true positives and true negatives as compared to false positives and false negatives.

Table III: Parameters Assessment of Our Approach

Total Sample	True Positives	True Negatives	False Positives	False Negatives	Sensitivity	Accuracy %
100	67	23	5	6	0.9178	90
200	131	51	9	9	0.9357	91
300	197	81	10	12	0.9426	92.67
400	291	92	12	5	0.9831	95.75
500	383	89	16	12	0.9696	94.4
600	448	122	19	11	0.9760	95
700	496	179	15	10	0.9802	96.43
800	559	219	13	9	0.9841	97.25
900	616	249	21	14	0.9778	96.11
1000	717	265	10	8	0.9890	98.2

(3) **Detection Rate or Sensitivity:** Detection rate or sensitivity is the measurement of the fraction of attack pattern that is correctly detected and selected. It is the ratio of true positives over the sum of true positives and false negatives.

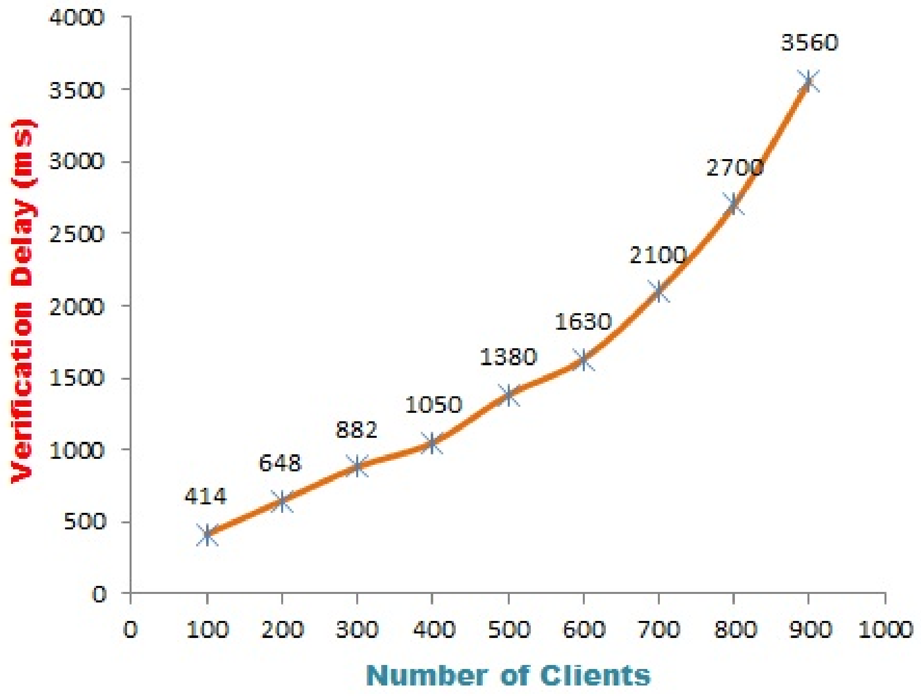

Figure 5: Verification Delay for Our Method

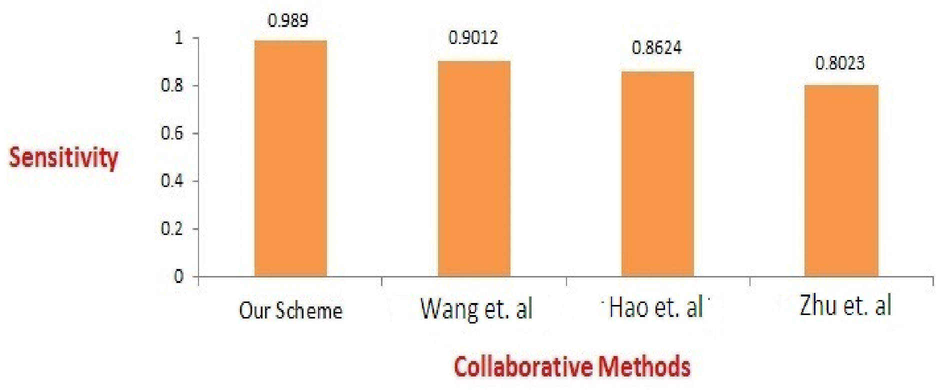

Figure 6: Sensitivity Comparison of Our Method with Other Collaborative Methods

$$Detection\ Rate\ (D_R) = Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{T_P}{T_P + F_N} \quad (10)$$

Where T_P = True Positives (Correctly selected) and F_N = False Negatives (Mistakenly Rejected).
For 1 TB file size, default HDFC file block size = 64 MB, n = 16384 blocks, r = 164 blocks, c = 460 blocks, total number of sample = 1000, audit frequency (Max 95Hz), and number of users =1000, we achieve 0.989 sensitivity which is comparatively higher than the existing

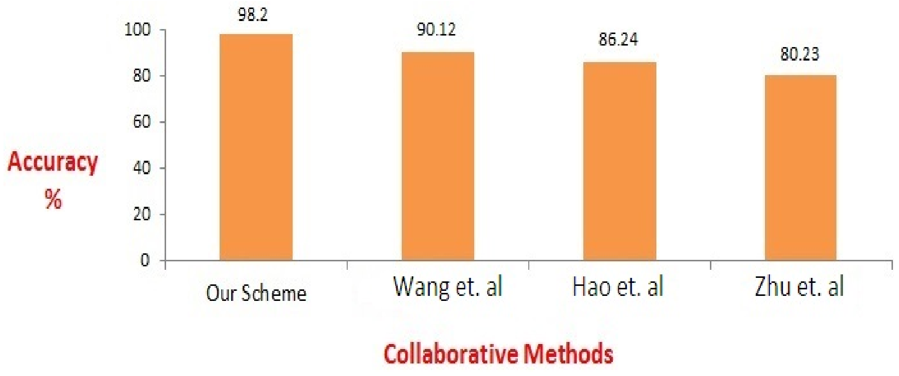

Figure 7 : Accuracy(%) Comparison of Our Method with Other Collaborative Methods

methods. The sensitivity results are shown in Figure 6.

(4) **Accuracy:** Accuracy or Classification Rate (C_R) is the ratio of true classified events(T_P+T_N) to the total number of actually occurred events ($T_P + F_P + T_N + F_P$).

$$Accuracy = Classification\ Rate(C_R) = \frac{T_P + T_N}{T_P + F_P + T_N + F_P} \tag{11}$$

Where, T_P = True Positives (Correctly selected), F_P = False Positives (Mistakenly Selected), T_N = True Negatives (Correctly Rejected) and F_N = False Negatives (Mistakenly Rejected).

$$Accuracy(in\ \%) = \frac{T_P + T_N}{T_P + F_P + T_N + F_P} \times 100 \tag{12}$$

With the same experimental configuration as described earlier, we achieve 98.9% accuracy which is comparatively higher than the existing methods. The accuracy results are shown in Figure 7.

6. CONCLUSIONS

In this paper, we propose a novel approach which uses PHC for data integrity verification. We suggested a fast variant of the traditional RSA scheme to speed up its decryption algorithm. Our RSA scheme uses a modulus of the form $N = p^r q$ where p; q are two distinct primes and $r \geqslant 2$. It employs the Hensel lifting and the Chinese remaindering to decrypt data. It provides the multiplicative homomorphism over the integers, and its security relies on the difficulty of both factoring the modulus N and finding the decryption exponent where the encryption exponent is private. The simulation results show that our RSA scheme offers good performance regarding running time, in comparison with the traditional RSA scheme while preserving a prescribed security level.

Ideally, the approach is suitable for cloud storage because of the advantages of PHC. Further, our technique supports dynamic data operations with less overhead. It also provides better security in case of Man In The Middle attack (MITM), Traffic flow analysis, Impersonation, Defacement, and misuse of data storage servers, because of the self-binding property of Paillier which can change cipher-text without any modification in plain-text and misguide the intruders.

Appendix

6.1 Preliminaries

We introduce some theorems and lemmas which are required to prove the correctness of the proposed scheme.

THEOREM 6.1. *(Chinese remaindering Theorem (X.Wang and Meng, 2015)). Let* $m_0; m_1; ....; m_{k-1}$ *be positive integers that are pairwise co- prime and let* $a_0; a_1; ....; a_{k-1}$ *be integers, then the set of the following congruences:*

$$x \equiv a_i \ (mod\ m_i) \tag{13}$$

Where $0 \leqslant i \leqslant k-1$ *has a unique solution.*

$$x \equiv a_0 M_0 M_0^{-1} + ... + a_{k-1} M_{k-1} M_{k-1}^{-1} \ (mod\ m) \tag{14}$$

Where $m = m_0 \times .... \times m_{k-1} = m_i M_i$ *and* $M_i M_i^{-1} \equiv 1 (mod\ m)$ *such that* $0 \leqslant i \leqslant k-1$.

THEOREM 6.2. *(The binomial theorem (Boneh et al., 1999)). Let* $n \in N$ *and* $x; y \in \mathbb{R}$, *then*

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} (x)^{n-k}.(y)^k \tag{15}$$

where $\binom{n}{k} = \frac{n!}{k!.(n-k)!}$

THEOREM 6.3. *(Fermats little theorem (X.Wang and Meng, 2015)). Let* p *be prime and* $x \in \mathbb{Z}$ *such that gcd(x,* p*) = 1, then*

$$(x)^{p-1} \equiv 1 (mod\ p) \tag{16}$$

LEMMA 6.4. *(Hensel lifting (Boneh et al., 1999)). Let* p *be prime and* $F(X) \in \mathbb{Z}[X]$. *Assume that*

$$F(X) \equiv G_1(X) F_1(X) (mod\ p) \tag{17}$$

Where $G_1(X)$ *and* $H_1(X)$ *are relatively prime in* $\mathbb{Z}_p[X]$. *Then for any integer* $r \succ 1$, *there exist polynomials* $G_r(X); H_r(X) \in \mathbb{Z}_p[X]$ *such that:*

$$F(X) \equiv G_r(X) F_r(X) (mod\ p^r) \tag{18}$$

Where $G_k(X) \equiv G_1(X)(mod\ p)$ *and* $H_k(X) \equiv H_1(X)(mod\ p)$

6.2 Paillier Homomorphic Cryptography System

We use a variant of PHC system (Paillier, 1999) for encryption and decryption. Group of integer numbers ($\mathbb{Z}_N$ and $\mathbb{Z}_N^*$) are utilized such that $\mathbb{Z}_N \times \mathbb{Z}_N^*$ is isomorphic to ${\mathbb{Z}_N^*}^2$. PHC system has three parts the key generation, the encryption, and the decryption algorithms which have described as below.

(1) **Key-Generation:**
In the first part, PHC system generates public and private keys for encryption and decryption, respectively a random number which is used to encrypt plain-text. It applies Euler's totient function on two different odd prime numbers to generate these keys. The procedure of key-generation is summarized as follows.
(a) An entity chooses two different odd prime numbers p and q of the same length.

(b) Calculate $J = pq$ and Euler's totient function on J is $\phi(J) = [(p-1)(q-1)]$.

(c) Assure that

i. $gcd(J,\ \phi(J)) = 1$.

ii. For any integer $a > 0$, we have $(1+J)^a = (1+aJ) \mod J^2$.

iii. As a consequence, the order of $(1+J) \in \mathbb{Z}^*_{J2}$ is J i.e. $(1+J)^J = (1 \mod J^2)$ and $(1+J)^a \neq (1 \mod J^2)$ for any $1 < a < J$.

(d) Selects a random $r \in \mathbb{Z}^*_J$ such that $gcd(L(r^J \mod J^2), J) = 1$, where $L(x) = (x-1)/J$.

(e) Return public key (J), the private key $(J,\ \phi(J))$ and a random number (r) of the system.

(2) **Encryption**
Second part of PHC system encrypts plain text using public key. Let $m \in \mathbb{Z}_J$ be a plain-text to be encrypted and $r \in \mathbb{Z}^*_J$ is a random number. With the definition of isomorphism, cipher-text can be obtained by function f that maps plain text as:
$\mathbb{Z}_J \times \mathbb{Z}^*_J \longrightarrow \mathbb{Z}^*_{J2}$ and
$c = E(m\ modJ, r\ modJ) = f(m,r) = [(1+J)^m \dot{r}^J \mod J^2\ ]$; where $c \in \mathbb{Z}^*_{J^2}$.

(3) **Decryption Algorithm**
In the last part of PHC system, the user can efficiently decrypt the encrypted text using its private key $(J, \phi(J))$. Decryption steps have given as follows.

(a) Set $\hat{c} := [c^{\phi(n)} \mod J^2]$ where c is cipher-text .

(b) Set $\hat{m} := (\hat{c}-1)/J$. (Note that all this is carried out over the integers.)

(c) After decryption, plain-text is given by $m := [\hat{m}.\phi(J)^{-1} \mod J^2]$

References

Ateniese, G., Burns, R. C., Curtmola, R., Herring, J., Kissner, L., Peterson, Z. N. J., and Song, D. X. 2007. Provable data possession at untrusted stores. In *ACM Conference on Computer and Communications Security.* 598–609.

Ateniese, G., Pietro, R. D., Mancini, L. V., and Tsudik, G. 2008. Scalable and efficient provable data possession. *IACR Cryptology ePrint Archive 2008*, 114.

Boneh, D., Durfee, G., and Howgrave-Graham, N. 1999. Factoring n= p r q for large r. In *Annual International Cryptology Conference.* Springer, 326–337.

Bowers, K. D., Juels, A., and Oprea, A. 2009. Proofs of retrievability: Theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security.* ACM, 43–54.

Chen, L. 2013. Using algebraic signatures to check data possession in cloud storage. *Future Generation Comp. Syst. 29,* 7, 1709–1715.

Cloudera. 2014. `http://www.cloudera.com/content/cloudera/en/downloads.html`. [Online; accessed 08-October-2014].

Dodis, Y., Vadhan, S. P., and Wichs, D. 2009. Proofs of retrievability via hardness amplification. In *TCC.* 109–127.

Erway, C. C., Küpçü, A., Papamanthou, C., and Tamassia, R. 2009. Dynamic provable data possession. In *ACM Conference on Computer and Communications Security.* 213–222.

Hao, Z., Zhong, S., and Yu, N. 2011. 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.

Juels, A., Bowers, K. D., and Oprea, A. 2009. Hail: a high-availability and integrity layer for cloud storage. In *ACM Conference on Computer and Communications Security.* 187–198.

Juels, A. and Jr., B. S. K. 2007. Pors: proofs of retrievability for large files. In *ACM Conference on Computer and Communications Security.* 584–597.

Lenstra Jr, H. W. 1987. Factoring integers with elliptic curves. *Annals of mathematics*, 649–673.

Paillier, P. 1999. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

Saxena, R. and Dey, S. 2014. Collaborative approach for data integrity verification in cloud computing. In *SNDS*. Communications in Computer and Information Science, vol. 420. Springer, 1–15.

Saxena, R. and Dey, S. 2016. Cloud audit: A data integrity verification approach for cloud computing. *Procedia Computer Science 89*, 142 – 151.

Saxena, R. and Dey, S. 2017. A curious collaborative approach for data integrity veri cation in cloud computing. *CSI Transactions on ICT 5,* 4, 407–418.

Sebé, F., Domingo-Ferrer, J., Martínez-Ballesté, A., Deswarte, Y., and Quisquater, J.-J. 2008. Efficient remote data possession checking in critical information infrastructures. *IEEE Trans. Knowl. Data Eng. 20,* 8, 1034–1038.

Shacham, H. and Waters, B. 2008. Compact proofs of retrievability. *IACR Cryptology ePrint Archive 2008*, 73.

Wang, C., Wang, Q., Ren, K., and Lou, W. 2010. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM*. 525–533.

Wang, Q., Wang, C., Li, J., Ren, K., and Lou, W. 2009. Enabling public verifiability and data dynamics for storage security in cloud computing. In *Computer Security–ESORICS 2009*. Springer, 355–370.

Wiener, M. J. 1990. Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information theory 36,* 3, 553–558.

Xenserver. 2014. `http://xenserver.org/discuss-virtualization/virtualization-blog.html`. [Online; accessed 08-October-2014].

X.Wang, G. Xu, M. and Meng, X. 2015. Mathematical foundations of public key cryptography. CRC Press.

Zhu, Y., Hu, H., Ahn, G.-J., and Yu, M. 2012. Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Trans. Parallel Distrib. Syst. 23,* 12, 2231–2244.

Rajat Saxena received the B.E degree from Jawaharlal Institute of Technology, Khargone and M.E degree from Shri Govindram Sakseria Institute of Technology and Science, Indore. He is working towards his P.hD under the supervision of Dr. Somnath Dey. He has ten years teaching experience in different colleges of India. His research focused on the computer network, security issues in Cloud computing and ad-hoc networks. Rajat is a member of IEEE computer society.

Somnath Dey is working as an Assistant Professor in the Discipline of Computer Science and Engineering at the Indian Institute of Technology, Indore. He has completed his B. Tech. from University of Kalyani in the year 2004 . He has earned an M.S. and P.hD from Indian Institute of Technology, Kharagpur in the year 2008 and 2013, respectively. His research focused on security issues in cloud computing, biometric security, image processing and human-computer interaction.