Data integrity verification: a novel approach for cloud computing

RAJAT SAXENA* and SOMNATH DEY

Cloud Computing Lab, Department of Computer Science and Engineering, Indian Institute of Technology Indore, Indore, India
e-mail: rajat.saxena@iiti.ac.in; somnathd@iiti.ac.in

MS received 5 October 2017; accepted 1 October 2018; published online 7 March 2019

Abstract. Data integrity verification (DIV) is one of the primary security checks for data stored on cloud servers. Cloud users have assured the safety of their data with frequent checking of data integrity. In this paper, we provide an efficient and effective DIV approach that offers effective and privacy-preserving audit structure. The main building components of our approach are a multi-power variant of the Paillier cryptography system with homomorphic tag. The Paillier cryptography system along with homomorphic tag assigns a unique and verifiable value to each data block, which helps performing dynamic data operations in cloud environment. To demonstrate our approach, we have implemented an application on Hadoop and MapReduce framework. We have tested this application with respect to the different parameters. The efficiency of the proposed method has been shown through the experimental results. Our method has shown a good improvement over those of the other modern systems.

Keywords. Homomorphic tag; paillier homomorphic cryptography (PHC); proof of retrievability (PoR); provable data possession (PDP); third-party auditing.

1. Introduction

Cloud computing [1–3] is the innovative on-demand automation of services to process, organize and store on remote cloud servers. Cloud environment [2, 4–6] suggests virtualization of resources to fulfill the service requirements of the cloud users. The Internet Protocol (IP) and other networking protocols are used to access the resources. Data integrity verification (DIV) [7, 8] is one of the massive responsibilities with cloud data, because the probability of involvement in malicious activities of a cloud user and cloud provider is very high. There are many ways to address this problem. The user can use encryption and decryption processes. However, it requires huge computing time and functional overheads. Data auditing methods may be the other way to address this problem.

Classical approaches for data auditing are Provable Data Possession (PDP) techniques [9–13] and Proof of Retrievability (PoR) techniques [14–18]. 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 afore-mentioned schemes are not suitable for batch auditing in the cloud environment because of their computation overhead. The benefits of these techniques depend only on the preprocessing steps that users can apply on outsource data file.

There is a settlement between dynamic data operations and privacy preservation. However, 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 [4, 19–21] that assign auditing tasks to a single Third-Party Auditor (TPA). TPA is an authentic and trusted entity that independently manages the data audits. The afore-mentioned schemes are not suitable for the batch auditing in cloud environment because of their computation and third-party overhead.

To address these problems, Saxena and Dey [7, 8, 22] have proposed multiple TPA schemes, in which various synchronous audit sessions from different users are handled by each single TPA simultaneously. We assume for our system that Cloud Service Provider (CSP) must be trustworthy.

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

1. We provide a public audit structure for privacy preservation and data storage security in the cloud environment.
2. Our approach supports dynamic data operations and provides an effective solution, which offers public auditing for the cloud environment.

*For correspondence

3. Proposed scheme can handle various efficient and synchronous audit sessions from multiple users.
4. Effectiveness of the proposed scheme has been demonstrated through implementations and experimentations.

Organization. We have organized the remaining paper into three sections. Section 2 will discuss our proposed scheme for DIV. In section 3, we describe the support of dynamic data operations. Security and performance analysis are presented in section 4. Finally, in section 5, we present the conclusions of our work.

2. Proposed scheme

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

2.1 Proposed DIV scheme

In this paper, we propose a two-tier architecture for privacy-preserving auditing. The first tier is cloud user and the second tier is CSP, which has two parts: (1) processing server and (2) encrypted data storage. The details of each tier are 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

2.2 Work flow of the proposed method

In this section, we present the process flow of our proposed audit scheme. In the 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 data to cloud user. Finally, cloud user decrypts the encrypted data. The workflow of the two-tier system architecture is described in figure 1. A detailed description of these tasks is given here.

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 a variant of the Paillier cryptography system [23]. Details of the Paillier cryptography system are given in Appendix 5.1. 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 $\lceil n/r + 1 \rceil$ bits long and use them to generate private and evaluation keys. We have chosen an integer e that is relatively prime to L and p . This integer is used to generate the private key. The process for generation of a key pair is 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 $d \equiv (e^{-1}) \pmod{L}$.
 - 3: Compute $d_p \equiv (d) \pmod{p}$ and $d_q \equiv (d) \pmod{q}$.
 - 4: Compute $k_p = \frac{N}{P_r} (\frac{N}{P_r})^{-1}$ where $k_p \equiv (1) \pmod{p^r}$ and $k_q \equiv (1) \pmod{q}$.
 - 5: Output: the evaluation key is $e_k = (N)$; the private key is $p_k = (N, p, q, r, e, d_p, d_q, k_p, k_q)$.
-

resources and are not capable of performing computation-intensive jobs, such as different auditing tasks (privacy preserving and DIV).

2. CSP

CSPs are organizations that provide sufficient infrastructure and resources 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.

2. Encryption

The cloud user divides the file F into a number of file blocks $m_1, m_2, \dots, m_k$. The size of data blocks is 512 MB. The cloud user encrypts these blocks with its private key (p_k) using RSA technique [24] and Euler's totient function. After this the 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 plain text. The ciphertext c_i is computed as follows:

$$c_i \equiv (m_i^e) \pmod{N}. \quad (1)$$

3. Storage

At the CSP server, file blocks are presented in encrypted form as individual ciphertexts $(c_1, c_2, \dots, c_k)$. These

Figure 1. Our approach for privacy auditing.

encrypted blocks and the evaluation key (e_k) are 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 the overall ciphertext and $(c_1, c_2, \dots, c_k)$ are the ciphertexts of each block, then C is generated using Eq. (2):

$$C = Eval(e_k, c_1, c_2, \dots, c_k). \quad (2)$$

The processing server needs to retrieve and verify the encrypted data $(c_1, c_2, \dots, c_k)$. To do this, CSP will provide overall individual c_i information given in Eq. (2). Now, ciphertext of each block is computed using m_i^e . It recomputes the cross product of individual block information $(m_1 \times m_2 \times \dots \times m_k)^e \bmod N$ using the

evaluation key e_k . Now, CSP will be able to evaluate the whole ciphertext C .

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

6. Response

The cloud service provider returns the processed result to the cloud user that is in encrypted form. With the help of the decryption method, the user can evaluate and verify the plain text M .

7. Decryption

For the given private key p_k , users utilize the Hensel lifting to decrypt the ciphertext C and construct a plain text. For this operation, modulus of (M_p^e) is conducted with respect to p^r . It is done for all values from 1 to $r - 1$. After this, the user performs the Chinese remaindering to recover the plain text. To do this, the user calculates $(M_p) \bmod p^r$ and $(M_q) \bmod q$ that alternatively generates M . Details of the Hensel lifting theorem and Chinese remaindering theorem are described in Appendix.

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

2.3 Correctness proof of effective privacy preservation

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

We define the threat vector in this part. Here, we select $r \geq 2$ 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 $d_p \equiv (d) \pmod{p-1}$ and $d_q \equiv (d) \pmod{q-1}$; 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 Eq. (1), we have the ciphertext $C \equiv (M^e) \pmod{N}$ where $M \in \mathbb{Z}_N$. Let us decrypt C modulo p :

$$\begin{aligned}
 M_0 &\equiv (C^d) \pmod{p} \\
 &\equiv (C_0 + bp^d) \pmod{p} \quad \text{where } b \in \mathbb{Z}. \\
 &\equiv (C_0^{d_p + a_1(p-1)}) \pmod{p} \\
 &\equiv (C_0^{d_p}) \pmod{p} \quad \text{thanks to theorem 3.}
 \end{aligned}$$

Similarly, we have, $M_q \equiv (C_q^{d_q}) \pmod{q}$.

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

Let the function $A_i(M_0, M_1, \dots, M_i)$ be defined as follows:

$$\begin{aligned}
 C_p &\equiv A_i(M_0, M_1, \dots, M_i) \equiv (M_0 + pM_1 + p^2M_2 \\
 &\quad + \dots + p^iM_i)^e \quad \text{where } 1 \leq i \leq r-1.
 \end{aligned}$$

Let us reduce C_p modulo p^{i+1} using binomial theorem 2:

$$\begin{aligned}
 A_i(M_0, M_1, \dots, M_i) &\equiv (M_0 + pM_1 + p^2M_2 + \dots + p^iM_i)^e \pmod{p^{i+1}} \\
 &\equiv \sum_{k=0}^e \binom{e}{k} (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^{(e-k)} \\
 &\quad (p^iM_i)^k \pmod{p^{i+1}}.
 \end{aligned}$$

We can explore this equation with the following expansion:

$$\begin{aligned}
 &\equiv (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^e \\
 &\quad + (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^{(e-1)} ep^iM_i \\
 &\quad + \sum_{k=2}^e \binom{e}{k} (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^{(e-k)} \\
 &\quad (p^iM_i)^k \pmod{p^{i+1}}.
 \end{aligned}$$

It can be divided into two subparts as follows:

$$\begin{aligned}
 &\equiv (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^e \\
 &\quad + ep^iM_i \sum_{j=0}^{e-1} \binom{e-1}{j} M_0^{e-1-j} (pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^j \pmod{p^{i+1}} \\
 &\equiv (M_0 + pM_1 + p^2M_2 + \dots + p^{i-1}M_{i-1})^e + p^i e M_0^{e-1} M_i \pmod{p^{i+1}}.
 \end{aligned}$$

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

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

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

$$M_i \equiv \left(\frac{C_p - A_{i-1} \pmod{p^{i+1}}}{p^i} \right) e^{-1} M_0^{1-e} \pmod p. \quad (3)$$

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

3. Support for dynamic data operations

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

3.1 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 $r_1, r_2 \in \mathbb{Z}_J^*$. The first homomorphic property says that the decryption of the product of two ciphertexts is equal to the sum of their corresponding plain texts. This can be represented by the following equation:

$$D[E(m_1, r_1)E(m_2, r_2)] = D[E(m_1 + m_2, r_1 r_2) \pmod{J^2}] = m_1 + m_2 \pmod J. \quad (4)$$

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

To update a file block m_i , users modify this block with a small change $\pm \Delta m_i$. It causes a new block $m_i \pm \Delta m_i$. Thus, change in ciphertext will be $(m_i \pm \Delta m_i)^e$. The final resulting ciphertext is $c_i \pm \Delta c_i$, after the commit operation.

3.2 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 is referred to 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 express this:

$$D[E^{m_2}(m_1, r_1)] = D[E(m_1 m_2, r_1^{m_2}) \pmod{J^2}] = m_1 m_2 \pmod J, \quad (5)$$

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

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

3.3 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 pre-determined 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.

3.4 Case study

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

1. Update operation

Cloud users request 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 Eq. (4). This property enables the server to update the file. A pictorial representation of update operation is given in figure 2. In figure 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 is described as $-\Delta c_{13}, \Delta c_{33}, \Delta c_{41}$ and Δc_{42} , then $c_{13}^*, c_{33}^*, c_{41}^*$ and c_{42}^* are depicted as updated output blocks.

2. Append operation

Cloud users request 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 Eqs. (5) and (6). This property enables the server to append the data to the file. A pictorial representation of

Figure 2. Explanation of update and append operation.

append operation is given in figure 2. In figure 2, we represent data blocks of a file as the C matrix 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^* . A description of append operation is given in figure 2.

3. Delete operation

Cloud users request CSP for the delete operation in a file. In partial delete operation, only some partial block information is deleted, and blocks are converted into modified blocks. The modified blocks coexist after the commit operation on blocks. We represent data blocks of a file as the C matrix. An explanation of delete operation is shown in figure 3.

In figure 3, three blocks c_{11} , c_{13} and c_{42} are selected for the partial delete operation, and $-\Delta c_{11}$, $-\Delta c_{13}$ and $-\Delta c_{42}$ are 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 fresh data blocks into the

scheduler, which can be assigned for storing the information.

In figure 3, two blocks c_{23} and c_{33} are selected for the full delete operation; $-c_{23}$ and $-c_{33}$ are the block changes; then the modified blocks are represented by null blocks as shown in figure 3, which can be assigned to the scheduler as fresh data blocks.

4. Analysis

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

4.1 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

Figure 3. Explanation of delete operation.

exponents, where the encryption exponent is also private. The fastest factorization algorithms, e.g., the Number Field Sieve and the Elliptic Curve Method [25], 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 $ed \equiv 1 \pmod{(p-1)(q-1)}$ 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:

$$[E(m_1, r_1)r_2^J] \bmod J^2 = E(m_1, r_1 r_2). \quad (7)$$

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

4.2 Performance analysis

We have executed our experiments with the set-up given in table 1. We have configured a cloud environment to perform auditing task of the stored files of the cloud users.

For the implementation, we have followed the recommendations for generating a key pair in [24, 28, 29]. We have set three moduli N_1 , N_2 and N_3 , each 1024 bits long formed of distinct primes as shown in table 2. Then, we have picked an encryption exponent $e = 216 + 1$. The implementation is shown in table 2, which presents 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.

The results of decryption time performance show that our RSA scheme gets a decryption speed-up by the factor of 2.84 on average over the traditional RSA scheme. However, using the RSA modulus formed from three distinct primes and employing the Chinese remaindering for decryption, cloud user gets a decryption speed-up by the

Table 2. Decryption time performance.

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

factor of 2.35 on average over the traditional RSA scheme. It seems certain that our RSA scheme gives a significant speed-up. The running of multi-power RSA decryption is nearly stable, thanks to the relationships of the Hensel lifting and the Chinese remaindering steps.

The best option to analyse the performance of any method is observing the performance parameters. We have measured the experimental results as regards four performance parameters and compared to those of other methods. The observations of different parameters are given in the following.

1. Detection probability of server misbehaviour

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, using the following equation:

$$P_X = P\{X \geq 1\} = 1 - P\{X = 0\} \\ = 1 - \left\{ \frac{n-r}{n} \cdot \frac{n-1-r}{n-1} \cdot \frac{n-2-r}{n-2} \cdots \frac{n-c+1-r}{n-c+1} \right\}. \quad (8)$$

P_X indicates the detection probability of server misbehaviour, which depends on total number of file blocks n , deleted blocks r and challenged blocks c . If the storage server deletes r blocks of the file, then the CSP detects server misbehaviour 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 samples = 1000, audit frequency (max 95 Hz) and number of users = 1000, we find results compared to those of other methods, which is shown in figure 4.

2. Verification delay

Verification delay is the time difference between the time of verification completion and the time of client

Table 1. Experimental set-up.

No of PCs	2
Processor	Intel core i7-2600S 2.80 GHz
Memory	16 GB
File storage server	Citrix Xen Server 6.2.0 [26]
Cloud environment	Cloudera CDH 5.3.0-0 [27]
Maximum data for storage	1 TB

Figure 4. Detection probability of server misbehaviour for our method and other techniques.

Figure 5. Verification delay for our method.

Table 3. 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

Figure 6. Sensitivity comparison of our method with other collaborative methods.

request for verification. It is a major factor for the verification of any data.

$$\text{Verification delay} = \text{time of verification completion} - \text{time of client request for verification.}$$

(9)

We compute the verification delay with the same configuration as described earlier and find comparatively

lesser verification delay, which is shown in figure 5. The time complexity of our experiments to calculate verification delay is $O(ck)$, where c is the number of required blocks for verification and k is the number of users.

Table 3 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 with false positives and false negatives.

Figure 7. Accuracy (%) comparison of our method with other collaborative methods.

3. Detection rate or sensitivity

Detection rate or sensitivity is the fraction of attack pattern that is correctly detected and selected. It is the ratio of true positives to the sum of true positives and false negatives.

$$\begin{aligned} \text{Detection rate } D_R &= \text{sensitivity} \\ &= \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \\ &= \frac{T_P}{T_P + F_N} \end{aligned} \quad (10)$$

where T_P = true positives (correctly selected) and F_N = false negatives (mistakenly rejected).

For 1 TB file size, default HDFS file block size = 64 MB, $n = 16384$ blocks, $r = 164$ blocks, $c = 460$ blocks, total number of samples = 1000, audit frequency (max 95 Hz) and number of users = 1000, we achieve 0.989 sensitivity, which is comparatively higher than those of the existing 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 ($T_P + F_P + T_N + F_N$):

$$\begin{aligned} \text{accuracy} &= \text{classification rate } C_R \\ &= \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \end{aligned} \quad (11)$$

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).

$$\text{Accuracy (in\%)} = \frac{T_P + T_N}{T_P + F_P + T_N + F_N} \times 100. \quad (12)$$

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

5. Conclusions

In this paper, we proposed a novel approach that uses Paillier homomorphic cryptography (PHC) for DIV. 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 $r \geq 2$. 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 when 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 ciphertext without any modification in plain text and misguide the intruders.

Acknowledgements

Authors would like to thanks the Computer Science and Engineering Department of Indian Institute of Technology Indore for their financial and infrastructure support.

Author Contributions Rajat Saxena (RS) and Somnath Dey (SD) contributed to this novel approach for the DIV in cloud computing. They designed the paper structure and gave their feedbacks to all its versions. SD designed the architecture of the described system and RS was responsible for implementation and testing. RS was responsible for the implementation of this work in the Hadoop and MapReduce project. This is in the development of context that provides the solution of the problem presented in this paper. Both authors approved the final manuscript.

Funding The work that led to this publication was partly funded by the government organization “Indian Institute of Technology Indore”.

Availability of data and materials The data and materials are available in our database upon any user’s request.

Compliance with ethical standards

Competing interests The authors declare that they have no competing interests.

Appendix

Preliminaries

Appendix1

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

Theorem 1 (Chinese remaindering theorem [5].) *Let $m_0, m_1, \dots, m_{k-1}$ be positive integers that are pairwise co-prime and let $a_0, a_1, \dots, a_{k-1}$ be integers; then the set of the congruences*

$$x \equiv a_i \pmod{m_i} \quad (13)$$

where $0 \leq i \leq k-1$ has a unique solution

$$x \equiv a_0 M_0 M_0^{-1} + \cdots + a_{k-1} M_{k-1} M_{k-1}^{-1} \pmod{m} \quad (14)$$

where $m = m_0 \times \cdots \times m_{k-1} = m_i M_i$ and $M_i M_i^{-1} \equiv 1 \pmod{m}$ such that $0 \leq i \leq k-1$.

Theorem 2 (The binomial theorem [6].) *Let $n \in \mathbb{N}$ and $x, y \in \mathbb{R}$. Then*

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

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

Theorem 3 (Fermat's little theorem [5].) *Let p be a prime and $x \in \mathbb{Z}$ such that $\gcd(x, p) = 1$. Then*

$$(x)^{p-1} \equiv 1 \pmod{p}. \quad (16)$$

Lemma 1 (Hensel lifting [30].) *Let p be a prime and $F(X) \in \mathbb{Z}[X]$. Assume that*

$$F(X) \equiv G_1(X) F_1(X) \pmod{p} \quad (17)$$

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

$$F(X) \equiv G_r(X) F_r(X) \pmod{p^r} \quad (18)$$

where $G_k(X) \equiv G_1(X) \pmod{p}$ and $H_k(X) \equiv H_1(X) \pmod{p}$.

5.1 Paillier homomorphic cryptography system

We use a variant of PHC system [23] for encryption and decryption. Groups of integer numbers ($\mathbb{Z}_N$ and $\mathbb{Z}_N^*$) are utilized such that $\mathbb{Z}_N \times \mathbb{Z}_N^*$ is isomorphic to $\mathbb{Z}_N^{*2}$. The PHC system has three parts: the key generation, the encryption and the decryption algorithms, which are described here..

1. Key generation

In the first part, the PHC system generates public and private keys for encryption and decryption, respectively; a random number 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.

- An entity chooses two different odd prime numbers p and q of the same length.
- Calculate $J = pq$ and Euler's totient function on J , $\phi(J) = [(p-1)(q-1)]$.
- Assure that
 - $\gcd(J, \phi(J)) = 1$.

ii.

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

iii.

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

- Select a random $r \in \mathbb{Z}_J^*$ such that $\gcd(L(r^J \pmod{J^2}), J) = 1$, where $L(x) = (x-1)/J$.
- Return public key (J) , the private key $(J, \phi(J))$ and a random number (r) of the system.

2. Encryption

The second part of PHC system encrypts plain text using public keys. Let $m \in \mathbb{Z}_J$ be a plain text to be encrypted and $r \in \mathbb{Z}_J^*$ be a random number. With the definition of isomorphism, ciphertext can be obtained by a function f that maps plain text as

$$\mathbb{Z}_J \times \mathbb{Z}_J^* \longrightarrow \mathbb{Z}_J^*$$

$$c = E(m \pmod{J}, r \pmod{J}) = f(m, r) = [(1+J)^m r^J \pmod{J^2}], \text{ where } c \in \mathbb{Z}_J^*.$$

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 are given as follows.

- Set $\hat{c} := [c^{\phi(n)} \pmod{J^2}]$ where c is ciphertext.
- Set $\hat{m} := (\hat{c} - 1)/J$. (Note that all this is carried out over the integers.)
- After decryption, plain text is given by $m := [\hat{m} \phi(J)^{-1} \pmod{J^2}]$.

List of abbreviations

F and $F[i]$	a file and the i^{th} data block of F
$E(\bullet)$ and $D(\bullet)$	the encryption and decryption algorithms
$H(\bullet)$	a hash function
$\phi(\bullet)$	Euler's totient function
$\mu(\bullet)$	a pseudo-random function (PRF) that maps $\mu : \{0, 1\}^k \times \{0, 1\}^l \longrightarrow \{0, 1\}^l$
$\sigma(\bullet)$	a pseudo-random permutation (PRP) that maps $\sigma : \{0, 1\}^k \times \{0, 1, \dots, n\} \longrightarrow \{0, 1, \dots, n\}$
p and q	two different odd prime numbers of the same length
J	multiplication of two prime numbers p and q
r_1, r_2	random numbers selected from the Galois field
T and T_i	all block tags and i^{th} tag of T
v	number of verification

R	number of blocks required for each challenge
k_t	encryption key for tag
k_d	decryption key for tag
$\mathbb{Z}$ and $\mathbb{Z}^*$	groups on integer numbers
c	number of blocks chosen for challenge operation
r	number of blocks deleted from total file blocks

References

- [1] Saxena R and Dey S 2015 Cloud shield: effective solution for DDoS in cloud. In: *Proceedings of Internet and Distributed Computing Systems – 8th International Conference, IDCS 2015*, Windsor, UK, September 2–4, *LNCS Proceedings*, 2015, pp. 3–10
- [2] Ruj S and Saxena R 2015 Securing cloud data. In: *Cloud computing with e-science applications*, ISBN:978-1-4665-9115-8. CRC Press, pp. 41–72
- [3] Lenstra Arjen K 1993 The number field sieve. In: *The development of the number field sieve*. Berlin–Heidelberg: Springer, pp. 11–42
- [4] Wang Q, Wang C, Li J, Ren K and Wenjing L 2009 Enabling public verifiability and data dynamics for storage security in cloud computing. In: *Proceedings of Computer Security – ESORICS 2009*. Springer, pp. 355–370
- [5] Wang X, Xu G, Wang M and Meng X 2015 *Mathematical foundations of public key cryptography*. CRC Press
- [6] Colin McGregor W S and Nimmo J 2010 *Fundamentals of university mathematics*, 3rd ed. Woodhead Publishing in Mathematics
- [7] Saxena R and Dey S 2016 Cloud audit: a data integrity verification approach for cloud computing. *Procedia Computer Science* 89: 142–151 (*Proceedings of the Twelfth International Conference on Communication Networks, ICCN 2016*, August 19–21, Bangalore, India)
- [8] Saxena R and Dey S 2014 Collaborative approach for data integrity verification in cloud computing. In: *Proceedings of the Second International Conference, SNDS*, Trivandrum, India, March 13–14, *CICS Proceedings*, 2014, pp. 1–15
- [9] Ateniese G, Burns R C, Curtmola R, Herring J, Kissner L, Peterson N J and Song D X 2007 Provable data possession at untrusted stores. In: *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 598–609
- [10] Ateniese G, Pietro R D, Mancini L V and Tsudik G 2008 *Scalable and efficient provable data possession*. IACR Cryptology ePrint Archive 114
- [11] Sebé F, -Ferrer J D, Martínez-Ballesté A, Deswarte Y and Quisquater J 2008 Efficient remote data possession checking in critical information infrastructures. *IEEE Transactions on Knowledge and Data Engineering* 20(8): 1034–1038
- [12] Erway C C, Küpçü A, Papamanthou C and Tamassia R 2009 Dynamic provable data possession. In: *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 213–222
- [13] Chen L 2013 Using algebraic signatures to check data possession in cloud storage. *Future Generation Computer System* 29(7): 1709–1715
- [14] Juels A, Kaliski B S 2007 PORs: proofs of retrievability for large files. In: *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 584–597
- [15] Shacham H and Waters B 2008 *Compact proofs of retrievability*. IACR Cryptology ePrint Archive 73
- [16] Dodis Y, Vadhan S and Wichs D 2009 Proofs of retrievability via hardness amplification. In: *Proceedings of the 6th Theory of Cryptography Conference*, San Francisco, CA, USA, March 15–17, pp. 109–127
- [17] Bowers D, Juels A and Oprea A 2009 Proofs of retrievability: theory and implementation. In: *Proceedings of the ACM Workshop on Cloud Computing Security*, ACM, pp. 43–54
- [18] Juels A, Bower K D and Oprea A 2009 HAIL: a high-availability and integrity layer for cloud storage. In: *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 187–198
- [19] Wang C, Wang Q, Ren K and Lou W 2010 Privacy-preserving public auditing for data storage security in cloud computing. In: *Proceedings of IEEE INFOCOM*, pp. 525–533
- [20] 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 Data and Engineering* 23(9): 1432–1437
- [21] Zhu Y, Hu H, Ahn G and Yu M 2012 Cooperative provable data possession for integrity verification in multicloud storage. *IEEE Transactions on Parallel and Distributed Systems* 23(12): 2231–2244
- [22] Saxena R and Dey S 2017 A curious collaborative approach for data integrity verification in cloud computing. *CSI Transactions on ICT*, doi <https://doi.org/10.1007/s40012-017-0172-x>
- [23] Paillier P 1999 Public-key cryptosystems based on composite degree residuosity classes. In: Stern J (Ed.) *Advances in Cryptology, EUROCRYPT 99*, Lecture Notes in Computer Science, vol. 1592. Berlin–Heidelberg: Springer, pp. 223–238
- [24] Boneh D, Durfee G and Howgrave-Graham N 1999 Factoring $n = p^r q$ for large r . In: *Proceedings of the Annual International Cryptology Conference*, August. Berlin–Heidelberg: Springer, pp. 326–337
- [25] Lenstra H 1987 Factoring integers with elliptic curves. *Annals of Mathematics* 126: 649–673
- [26] XenServer 2014 *Download xenserver 6.2*, <http://xenserver.org/open-source-virtualization-download.html>
- [27] Cloudera 2014 *Cloudera downloads get started with Hadoop*, <http://www.cloudera.com/content/cloudera/en/downloads.html>
- [28] Wiener M 1990 Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information Theory* 36(3): 553–558
- [29] ANSI Standard X9.31 1998 *Digital signatures using reversible public key cryptography for the financial services industry (rDSA)*
- [30] Yun D Y 1974 *The Hensel lemma in algebraic manipulation*. No. MAC-TR-138, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC