Security For Dynamic Data Storage in Cloud

P.VEERALAKSHMI
Department of Computer Science and Engineering
B.S.S Abdur Rahman University
Vandalur- Chennai
INDIA
veerphd1@gmail.com
LATHA TAMILSELVAN
Department of Information Technology
B.S.S Abdur Rahman University
Vandalur- Chennai
INDIA
latha_tamilnselvan@yahoo.com

Abstract:- Cloud provides variety of services and resources dynamically through internet. In cloud, the users avail the storage space provided by cloud service providers. In this , the users do not have direct control over the remotely stored data. The imperative security concerns in cloud are the integrity and confidentiality of data. To preserve the privacy of dynamically changing data, an efficient approach maintaining the confidentiality and assuring the integrity of data is proposed. In this scheme, public auditability is enabled by introducing a third party auditor . We ensure that the data stored in the untrusted cloud server is confidential and consistent by 2-keys Symmetric Encryption. Unlike other encryption algorithms, this needs lesser computation overhead. Simulation environment is set up with the eucalyptus tool. The performance analysis and simulation results prove that our proposed scheme is secure and proficient by reducing the computation cost of server and user.

Key-Words:- Dynamic data Operations, Public Verifiability, Symmetric Encryption, Confidentiality, Remote Integrity, Data Storage Security, Privacy Preserving Auditing

1 Introduction

Cloud computing is used for the delivery of hosted services over the Internet. It enables organizations to consume computing as an utility – similar to electricity or a telephone service – rather than building and maintaining, computing infrastructures. The users can access data and applications from remote servers with the fixed or mobile devices. Storage becomes an increasing attraction in cloud computing paradigm. Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Services(S3) are the well-known cloud storage services. With the cloud, the small organizations can hire the resources from the cloud rather than puchasing them and avoid the capital costs for software and hardware. With cloud , IT infrastructure can be easily adjusted to accommodate the changes in demand .

There are several [6] issues like security, resource scheduling, memory management,data migration, data privacy, access control,etc., Cloud providers move the users' applications, softwares and databases to the large data centers that are located anywhere in the world. This distinct feature of cloud reports many new security challenges like confidentiality and integrity.At the same time,the cloud offers many benefits like enhanced collaboration, portability, limitless flexibility,etc. To enjoy the full benefits of cloud , the users have to store their data in the encrypted format.

Encryption of data can handle the confidentiality issue. But verification of integrity without having a local copy of data is a difficult task in cloud. Some of the existing methods like SHA,MD5 [18] can not be directly used. The simplest way to check the integrity of data is to download the full data stored in the cloud to ensure its integrity. It incurs excessive I/O cost and heavy communication overhead across the network. So some other effective method is required for assuring the confidentiality and intgerity of data stored in the cloud with the minimum overhead.

Recently many remote integrity checking methods [3][4][5][7][8][9] were proposed to check the integrity of data stored at the remote server. In these some of the methods are not dealing with confidentiality and are not supporting the dynamic data operations. So, a new cryptographic mechanism for protecting confidentiality and integrity of stored data in cloud is needed.

- *Confidentiality*: It ensures that computer information are used and gained access by only authenticated and authorized individuals.
- *Integrity*: It denotes that the data in the cloud can be updated only by authorized users. Updates of data file include writing ,appending to the existing data, changing, and deletion.

Since, the existing encryption algorithms cannot be used for the encryption of bulk data, we are proposing a new symmetric encryption algorithm for maintain confidentiality and integrity.

2 Related Work

2.1 Security Concerns in Cloud

There are various security issues in cloud because the users are not having direct control over the data stored in cloud. Jensen [10] deliberated the security issues arising from the usage of cloud services and by the technologies used to build the internet-connected and cross-domain collaborations. It emphases on browser security, cloud integrity , WS-security, transport layer security and binding issues in the field of cloud.

2.2 Merkle Hash Tree(MHT)

The verification of correctness of data stored in cloud server by allowing the third party auditor was discussed by Wang[2] . With the aid of Merkle hash tree it is possible for the clients to preserve the level of data correctness assurance by performing block-level operations on the data files. In this scenario , chances are there for the third party auditor to abuse the data while undertaking the verification process. Lifei Wei et al.,[11] established a new mechanism to verify the precision of computations (addition,subtraction, multiplication, division, etc.) done by the cloud service provider. For that, they have used the Merkle hash tree for checking the computation accuracy . The only criteria is the number of computations submitted to the server for processing must be in the order of power of 2, since the Merkle hash tree has the 2^n number of leaves.

2.3 Advance computation of Tokens

A storage correctness model for substantiating the accuracy of stored data by computing a few number of tokens was proposed by Wang et al.,[12] This insists the user to pre-compute a number of verification tokens, each covering arbitrary set of data blocks. It allows the cloud user to challenge the cloud server with a set of pre-computed tokens. After getting the challenge token, the cloud server computes a signature over the specified data blocks and returns the signature to the cloud user. The signatures that are returned by the provider should match the appropriate tokens pre-computed by the user. The main challenge of this system is that the cloud user can able to test the cloud server only for a definite number of times.

2.4 Proovable Data Possession and Proof Of Retrievability Scheme

For checking the integrity of stored data, some sentinel characters were embedded in the data file by A.Juels and B. S. Kaliski,[7]. These sentinels were hidden in the data blocks . During verification , the user can challenge the server by revealing the positions of sentinels and demands the provider to return the relevant sentinel vlaues. This procedure allows the user to challenge the cloud server for a limited number of times by deliberating the positions of the sentinel values in advance. G.Ateniese et al.,[8] proposed a model called “Provable Data Possession” to ensure the possession of files stored on the untrusted server. They used RSA- based homomorphic tags for evaluating the correctness of outsourced data. Here also the user needs to pre-compute the tags and stores all the tags in advance . The computation of tags requires a lot of computation overhead and storage space. The homomorphic properties were also used to check the integrity of data [13]. For ensuring the remote integrity, the MAC and reed solomon code were used [14]

2.5 Dynamic Data Operations

Many of the existing remote checking methods support only static data [3][7][8][13][14]. These are not featured with the methods for handling dynamically changing data. Several methods have been proposed for provisioning dynamic data in cloud [4][9][15][16][17]. In these, some of the papers are not posing the support for block insertion operations and are detecting the data corruptions with a lesser probability [4]. For achieving high probability of detection of data corruptions ,large number of challenges are required for the server from the client or TPA. And also, these methods are not considering the issue of confidentiality. In this paper, we are proposing a method assuring both confidentiality and integrity of dynamically changing data.

Our scheme presents a stream cipher encryption algorithm called 2-keys symmetric encryption for protecting the confidentiality of data. This method generates the metadata for all the data blocks stored in the server for ensuring the integrity of data.

2.6 Secure storage and secure computation

To ensure the integrity of stored data in cloud, a scheme considering the positions of data has been suggested in [3]. And to ensure secure computation, this method uses the Merkle hash tree for checking the correctness of computations done by the cloud service provider. This method is not featured for the dynamically changing data stored in cloud.

3 Problem Definition

The major security issues in cloud computing are integrity and confidentiality of data stored at the servers. In cloud, the service providers and consumers should ensure that the data stored at the server is secure. In this paper a method for ensuring data integrity and confidentiality for dynamically changing data has been proposed . Dynamic data operations like insertion , deletion, modification and appending are conceivable without retrieving the entire data from server by using the linked list data structure. Here public auditablity is enabled by introducing a third party auditor (TPA) without disclosing original data to the TPA.

3.1 System Model

Our storage model consists of Data Owners(DO), n cloud storage servers(s1,s2,..sn) under the direct control of Cloud Service Providers (CSP) and Third Party Auditors(TPA). Storage servers provide storage services.

Data Owners: Users having their data to be stored in cloud and depending on the CSPs for data computation. The client can be the individual user or an organization.

Cloud Service Providers: It can be the organizations comprising many storage servers and offering significant storage space and computing resources.

Third Party Auditors: The Data Owner may call the third party auditor to verify the integrity of data. The verifier's role falls into two categories.

i. Private Auditability: It permits the Data Owner to verify the integrity of data file stored in the server.

ii. Public Auditability: It allows anyone including TPA to verify the intgerity of data stored in the server.

3.2 Threat Model

It is presumed that the TPA is honest. It executes honestly in the whole auditing process of checking the integrity of data. The data will not be leaked out to the third party auditor during the auditing process. But the server may conduct the following attacks:

Replace Attack: If the server discarded a challenged block or its metadata, it may choose another valid uncorrupted pair of data block and replace the original data block and metadata.

Replay Attack: The server generates the proof from the previous proof without retrieving the challenged data.

External Attacks: The external hackers who is capable of compromising the cloud servers can access the data stored in the server . He may delete or modify the data and may the leak the sensitive information.

3.3 System Overview

To ensure the confidentiality and integrity of data stored in cloud, an efficient scheme has been proposed. The overall architecture of our system is described in the Fig.1. This system consists of four phases namely Setup phase, Verification phase , Dynamic data operations and Decryption phase.

3.3.1 Setup Phase

The data owner(DO) preprocesses the file F before storing it in cloud server. The setup phase consists of four steps. 1. Key generation 2. Binary sequence generation 3.Encryption and 4. Metadata generation.

a. Key generation

It enables the computation of two Keys(key1 and key2) by using the ascii values of characters in the keywords and their position in the keyword.

The DO generates two symmetric keys by using the algorithm1 [1]. This algorithm takes two keywords(keyword1 and keyword2) from the user as the input.

Algorithm 1: Key generation
Input: keyword1, keyword2 **Output:** key1,key2

```
To generate key1
    1. Initialize the key1 as key1=0 and i=0.
    2. key1=key1+(ASCII(keyword1[i])*(i+1))
           i→the index of each character in the keyword1.
    3. Repeat the step 2 for all the characters in the keyword1.
To generate key2
    1. Initialize the key2 as key2=0 and i=0.
    2. Key2=key2+(ASCII(keyword2[i])*(i+1))
           i→the index of each character in the keyword1.
    3. Repeat the step 2 for all the characters in the keyword2.
```

b.Binary Sequence Generation

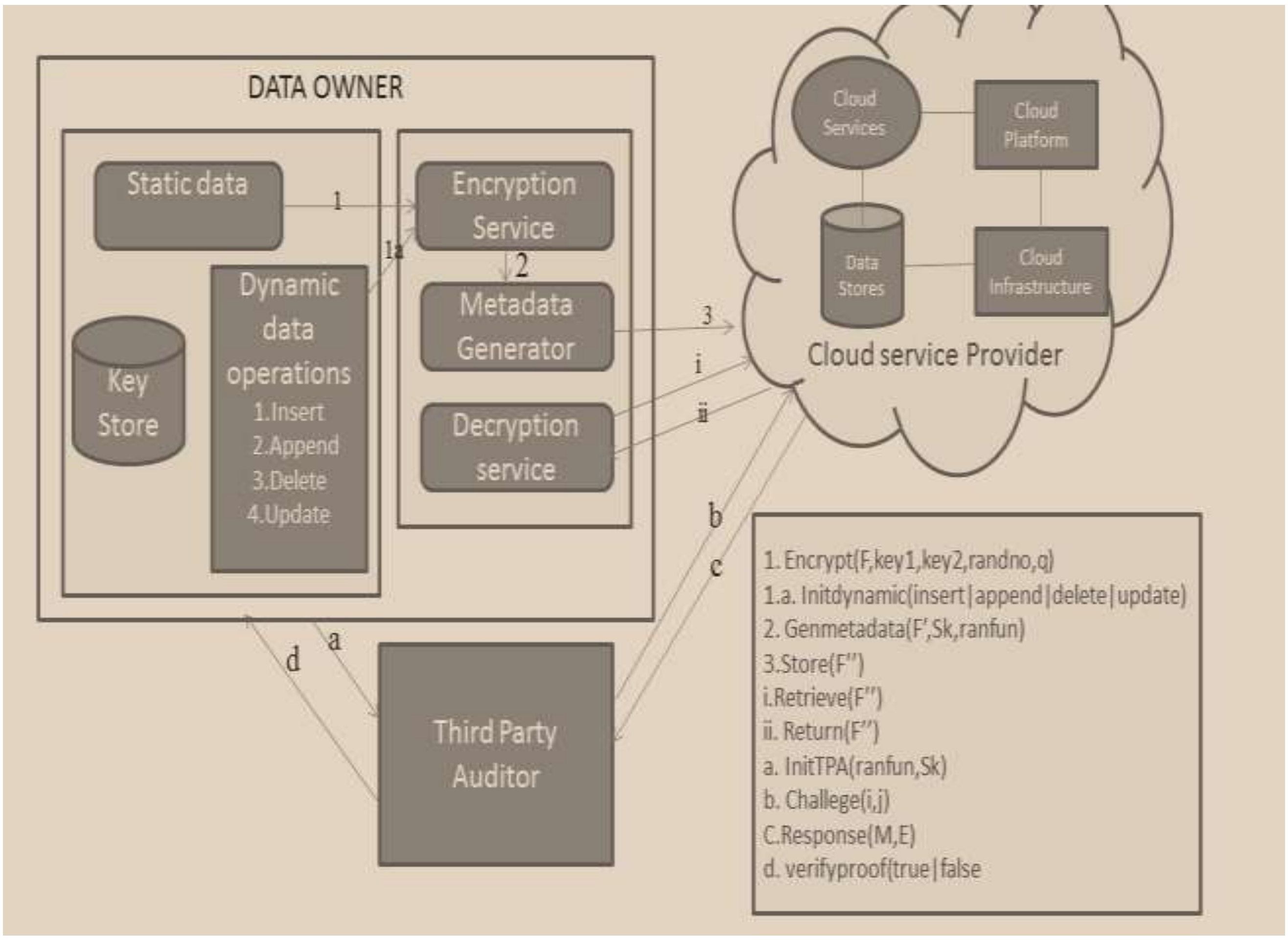
The DO generates the binary sequence consisting of 0s and 1s . This binary sequence is generated using the recurrence relation of the form as in the equation.1

$$X_{n+m}=(C_0X_n+C_1X_{n+1}+C_2X_{n+2}+\ldots\ldots C_{m-1}X_{n+m-1})\bmod 2 \qquad (1)$$

For generating the recurrence relation, the user needs an initial seed value m, the initial vector values like (X_1, X_2, X_3, X_4,.. X_m) and the coefficient values like (C_0,C_1,C_2 C_3,,..C_{m-1}).
For example, for the initial seed m=5 , initial vector (0,1,0,0,0) ie . $X_1=0$; $X_2=1$; $X_3=0$; $X_4=0$; $X_5=0$ and
for the coefficient (1,0,1,0,0) ie.$C_1=1$; $C_2=0$; $C_3=1$; $C_4=0$; $C_5=0$, m=5 the recurrence relation is like the equation.2.

$$X_{n+5}=X_n+X_{n+2} \qquad (2)$$

The binary sequence generated for the initial vector (0,1,0,0,0) and the coefficient (1,0,1,0,0) is 0100001001011001111100011011101010000 .



1. Encrypt(F,key1,key2,randno,q)
1.a. Initdynamic(insert|append|delete|update)
2. Genmetadata(F',Sk,ranfun)
3.Store(F'')
i.Retrieve(F'')
ii. Return(F'')
a. InitTPA(ranfun,Sk)
b. Challege(i,j)
C.Response(M,E)
d. verifyproof(true|false

Fig.1: System Architecture Diagram

c.Encryption

To ensure the confidentiality of data, the DO encrypts each data block using 2-keys symmetric encryption algorithm[1]. It takes as input the binary sequence, key1,key2 and a secret random number for encryption. This encryption algorithm is not only depending upon the keys, but also the position of the characters in the file.

Algorithm 2: 2-Keys Symmetric Encryption
Input: key1,key2, binary sequence,randno, File F
Output: Cipher Text

Split the data file **F** into **n** data blocks **db1, db2, db3,.. dbn.**

Cosider that each of the **n** data blocks contains **s** bytes .

1. Read the first bit in the binary sequence.
2. If the bit value is one , encrypt the block with the **key1** using the following equation.

Encrypt[i,j]=(key1+ASCII(F[i,j])+randno$^{(i+j)}$)mod 256

i→refers to the block number

j→index of the character in the ith block.

3. If the bit value is zero , encrypt the block with the **key2** using the following equation .

Encrypt[i,j]=(key2+ASCII(F[i,j])+randno$^{(i+j)}$)mod 256

4. Write the encrypted value(Encrypt[i,j]) in the file **F'**.
5. Repeat the steps 2 or 3 and 4 for all the characters in that block.
6. Read the next bit in the binary sequence.
7. Repeat the steps 3-7 for the datablocks in the text file.

---

d. Metadata generation

After encryption, the data owner computes the metadata of the encrypted data blocks to ensure the integrity of data stored in the server. To generate the metadata, the algorithm discussed by Ponnuramu Veeralakshmi. and Latha Tamilselvan[3] can be used.

It is making use of some random functions containing the location of charaters in the file **F'** and a secret key(Sk) chosen by the DO . The random function should be kept as secret one by the DO. The random function and the secret key(Sk) should be submitted to the TPA for confirming the integrity of data.

An example for such random function f(i,j) to generate the Metadata M is

f(i,j)=M[i,j]=ASCII(F[i,j])*i*j*Sk. (3)

The generated metadata using the equation.3 is concatenated with the encrypted data of each block and is named as **F''.** The file **F''** will be stored in the cloud server by the data owner.

3.3.2 Verification Phase

After storing data in the server, to ensure the integrity of data, our system depends on this phase. The Data Owner assigns this job to the third party auditor(TPA).

The DO submits {f,Sk} comprising of the random function, secret key used to generate metadata to the TPA. After receiving the key and the random function, the TPA creates a challenge message (chal) and sends it to the untrusted server. Upon receiving the challenge message from the TPA, the server generates a response message and send it to the TPA. Using this, the TPA checks the integrity of the message as discussed with [3].

3.3.3 Dynamic Data Operations

The proposed scheme provisions dynamic data operations at the block level such as Modification, Insertion and Deletion operations. To achieve the dynamic data operations, we can use indexing [4] and Merkle Hash Tree [5]. Here we are using the simple data structure called the linked list. When generating metadata for the data file F, create a singly linked list such that each node in the list consists of the starting position of the new block .The linked list for the file will be stored with the data owner along with the keys for encryption. The DO can use the linked list to retrieve the original data in the correct order. The TPA and cloud server (CS) are not knowing about the linked list maintained with the DO.

Construction of linked list

For a file F containing **n** blocks with **BS** as the block size, linked list can be constructed as the Fig.2. Each node contains the starting position of the new block. This linked list is maintained with the DO for retrieving the data in order.

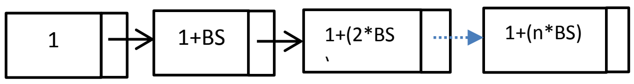


Fig.2: Linked List Construction

3.3.3.1 Block Insertion

In this , the Data Owner can insert a new block db* after the position **k** in the file **F''={db1,db2,db3,…dbn}**. Usually the insertion operation changes the logical structure of the file **F''**. So, instead of inserting the new block at the middle, we try to append the block at the end and insert a node at the position **k** in the linked list maintained with the DO. Using this tecnique, a new block can be inserted without re-computing the metadata and encrypted data for all blocks that have been shifted after inserting a block. It enables the calculation of the encrypted metadata only for the particular block which needs to be inserted. Block insertion can be done using the Algorithm.3.

Algorithm 3: Block Insertion
Input: Block to be inserted **{db*}**,Position **k**
Output: Appended data at the server,Inserted Linked list at the DO

```
1. Get the new block to be inserted. (ie) db* and the position k after which it is to be inserted.
2. Peform the encryption for the new data block db* using the 2-keys symmetric encryption algorithm as explained in algorithm.2 considering the position i , the end of data file F''.
3. Generate the metadata of the encrypted block.
4. Append the metadata and encrypted data the end of the file F'' in the server.
5. Insert a node in the linked list at the position k .The data at the node should be the position at which the data was appended in the server
```

The server contains the data as **{db1,db2,db3,..dbk-1,dbk, dbk+1,.. db*,}**
After inserting the data, the linked list maintained by the Data Owner will be updated as in the Fig.3.

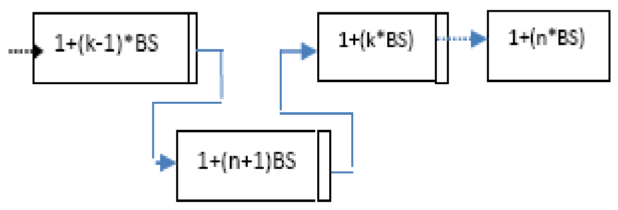


Fig.3: Linked list after Block insertion

3.3.3.2 Block Modification

Modification of data can be simply performed by using this proposed method. If the user wants to update a block **db2** at the position **k** with another block **dbm** ,then for the individual block **dbm,** the encrypted data as well as the metadata can be calculated without affecting the other blocks in the server. After calculating the encrypted metadata , the DO makes an update request like **Update(F'',dbm,k)** to the server to modify the block **db2** with **dbm**.

Upon receiving the update request **Update(F'',dbm,k)** , the server update the data as {**db1,dbm,db3,..dbn}.** It is not required to update the linked list in the user side for block modification operation.

3.3.3.3 Block Appending

Data can be appended to the data stored in the cloud server without retrieving the whole data from the server.To append a block **dba** at the end, the data owner computes the encrypted data and metadata for that block without disturbing the other blocks. Then the user makes an append request **Append(F'',dba)** to the cloud server to append the datablock **dba** at the end of the File **F''**.

The server appends the data block **dba** at the end as **{db1,db2,db3,..dbn, dba}**

The client updates the linked list as in the Fig.4.

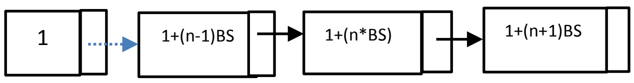


Fig.4: Linked list after Block Appending

3.3.3.4 Block Deletion

To delete a block at the position **k**, the user makes a delete request **Delete(F'',k)** to the server and the server deletes the encrypted data and the metadata at the position **k** and replaces the block with **null(Φ)**.For example to delete a block at the 2nd position ,the server updates the file as **{db1,Φ,db3,..dbn} .** The client updates the node at the 2nd position with **Φ.**

3.3.4 Decryption phase

With the help of the secret key**(Sk)** and the reverse of the random function **f(i,j)** used to generate the metadata, the user recovers the encrypted data from the metadata. From the encrypted data, the DO gets the original data using the decryption algorithm as explained in the algorithm.4

```
Algorithm 4: 2-Keys Symmetric Decryption
Input: key1,key2, binary sequence,randno, Encrypted File F''
Output: Original Data
```

The user generates the binary sequence from the recurrence relation, initial vector and the coefficients. The keys key1,and key2 are generated from the keywords keyword1,keyword2 respectively for decryption.

Divide the encrypted datafile encdata.txt into **n** data blocks **db1, db2, db3,.. dbn**

Consider each **n** data blocks contains **s** bytes like **b1, b2, b3,.. bs**

```
1. Read the first bit in the binary sequence.
2. If the bit value is one , decrypt the block with the key1 using the following equation.
   Decrypt[i,j]=(encdata[i,j]-key1)-randno^(i+j))mod256
      i→refers to the block number
      j→index of the character in the ith block
3. If the bit value is zero , decrypt the block with the key2 using the following equation.
   Decrypt[i,j]=(encdata[i,j]-key2)-randno^(i+j))mod256
      i→refers to the block number
      j→index of the character in the ith block.
4. If Decrypt[i,j]<0 then do the following
      Decrypt[i,j]=decrypt[i,j]+256
      i→refers to the block number
      j→index of the character in the ith block.
5. Write the value of decrypt[i,j] in the decryptdata.txt file
6. Read the next bit in the binary sequence.
7. Repeat the steps 3-7 for the datablocks in the text file.
```

4 Security Analysis

In this section,we present the security analysis for the integrity and confidentiality of the stored data in cloud.

4.1 Integrity

To assure the integrity, we need the following properties.

Public Verifiability: It permits anyone not just the DO to check the integrity of data . In our system, a third party auditor(TPA) is permitted for integrity verification. So, this supports the public verifiability and also the private verifiability.

Privacy of data: Because the verification is done at the TPA, we should assure the privacy of data such that no information should be leaked to the third party auditor(TPA). In our scheme, the TPA does the auditing only in the encrypted data whose keys are maintained with the DO. So it is guaranteed that, no information is leaked to the TPA.

Blockless verification: No challenged file blocks should be fully retrieved by the TPA during the verification phase for security concerns. In our system, the TPA gest only two characters(metadata and encrypted data) at the posisions specified by the DO. No full block is retrieved from the server. Our System ensures blockless verification.

Low Computation: Only a lesser computation should be done at the TPA to verify the integrity of the file.

Low Storage Overhead at TPA: The amount of storage in the TPA is as small as possible to check the integrity. In our scheme, the TPA has to store only the random function and the secret key (Sk). So the storage at the TPA is minimum.

Support Dynamic Operations: After storing the data in server, the user can update the data dynamically. It should able to perform dynamic operations like Block Insertion,Block Modification, Block Deletion and Block Appending. Our system supports efficiently supports all the dynamic operations.

Probability of detection of data corruption (Pd): The TPA should check the data corruption with high probability.

We investigated the probability **Pd** based on the the type of data corruptions done at the untrusted server. The data corruption can be classified into data deletion, data modification, data insertion and data appending. Using this method, data deletion,data insertion and data appending corruptions will be detected with high probability of 1 with minimum number of challenges, because these operations change the position of characters in the file F.

To find the probability detection(Pd) of data replacement corruption, consider the following assumptions.

- The file F contains the **n** data blocks and the probability to pick anyone of these blocks is **1/n.**
- The attacker modifies **m** blocks and the probability of modified block is **m/n.**
- The TPA makes **t** number of challenges to the server to detect the corruption.
- **s->** the number of bytes in a block.

Based on these assumptions , probability detection of data replacement corruption is calculated by using the equation. 4.

$$Pd=1-(1-m/n)^{ts} \quad (4)$$

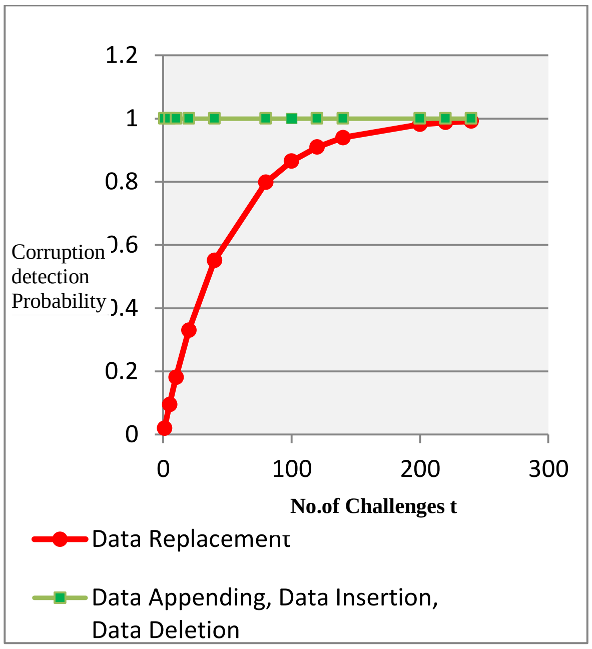


Fig.5: Probability of detection Pd of data corruptions for 1000 blocks, 20 sectors in a block and , 1corrupted block

The Fig.5 illustrates the probability detection of data corruptions like data replacements, data appending, data insertion and data deletion for the file containing n=100 blocks ,s=10 sectors in a block and 1 corrupted block. This figure infers that the data appending, data insertion and data deletion corruptions are acknowledged with the highest probability of 1 demanding minimum number of challenges. The probability of detection of data replacement corruptions depends on the number of challenges the TPA issues to the server for verifying the integrity of data.

4.2 Confidentiality

We have carefully analyzed the confidentiality issues of this method . Here, a symmetric stream cipher encryption algorithm has been proposed. Stream ciphers are suitable for the encryption of larger data than the block cipher methods. In block cipher, the data is split into several blocks of certain size. The same encryption algorithm and keys are used for the encryption of all the data blocks. If the plaintext contains some identical data blocks , we get the identical ciphertext blocks if electronic code book (ECB) mode is used. The modes of operations like Cipher Block Chaining(CBC), Cipher FeedBack(CFB) and Output FeedBack (OFB) are not suitable for dynamically changing data. So, it is decided that block ciphers are not suitable for larger dynamically changing data.

And also, the existing stream cipher tecniques are vulnerable to frequency analysis attack, brute force attack, correlation attack, algebraic attack, known plain text attack,cipher text only attack,etc. One time pad encryption algorithm is getting relaxed from these attacks. But for this, the key should be as lengthier as the plaintext. And to generate the key, it is required to use the LFSR sequence ,blub blub generator, or recurrence relations. These tecniques consumes more time to generate a longer binary sequence. And also the generated binary sequence is prone to correlation attacks. So, we have proposed the 2-keys symmetric encryption algorithm which is freed from all the attacks.

4.3 Analysis of 2-keys symmetric algorithm

Brute Force attack:

In this method, two keywords are used for encryption and also the keywords are of variable sizes. Along with the keywords , initial vector **values (X_1, X_2, X_3, X_4,.. X_m)**, the initial coefficients **(C_0,C_1,C_2C_3,,..C_{m-1}).**, random number **randno**, and the number of characters in the block **q** are the secret information maintained by the cloud user. So the chance of brute force attack is very less with this encryption algorithm.

Frequency Analysis Attack

This cryptanalysis method computes the number of occurrences of characters in the cipher text and plain text. And it also compares their frequency of occurrences. With this method, an experimental analysis was conducted for different strings to measure the level of security of data.

First, as a test case we chose, a file which contained the plaintext "aaaaaaaaabbbbbbbbbbcccccccccccddddddddddd deeeeeeeeeeeefffffff" and used this method to encrypt the data. Fig.6. shows the frequency of characters of the encrypted data. From this, it is inferred that, for the same characters of plain text, we get the different cipher text characters. This feature makes the frequency analysis attack nominal for the cryptanalyst.

Cipher text only attack:

It is the attack that cryptanalyst tries to get the plaintext and the key from the ciphertext. Since in this encryption , the same plain text character is encrypted to different characters , it is not conceivable to find the keys and the plain text.

Known plain text attack:

In this, the cryptanalyst got the part of plain text along with its cipher text. With these information, he tries to get the entire plain text and the key. Since in this encryption algortihm 2 different keys are used, eventhough a part of plaintext along with the cipher text is known to cryptanalyst, he can able to get only one key and not the second key. Even, if the key1 and key2 are hacked, the hacker has to get two more secret keys (ie) binary sequence and a random number.

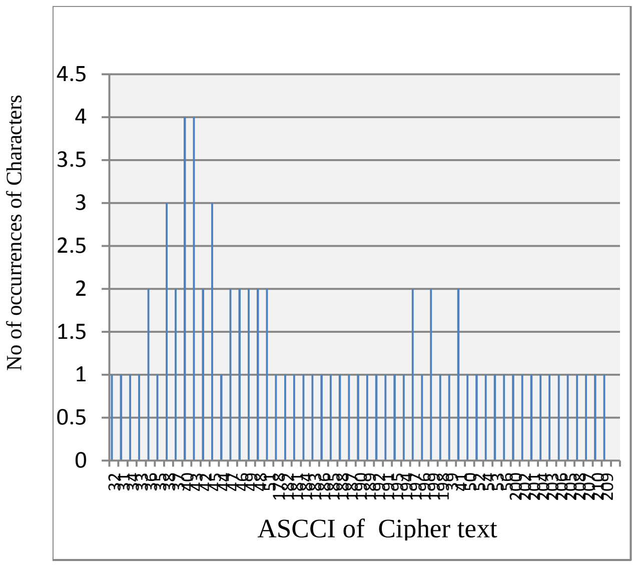


Fig.6: Frequency of characters of encrypted data

Correlation attacks:

The attacks try to extract some information about the initial state from the output stream. Here, since the recurrence ralation has been used to generate binary sequence, the attacker can try to get the initial seed (initial vector and initial coefficients) from the binary sequence. If the length of the initial seed is smaller, then the binary sequence will be getting repeated. Then it is possible for a cryptanalyst to get initial seed. In order to avoid this the user must choose a larger initial seed.

4.4 Complexity of Encryption Function:

The encryption function, we have proposed is

Encrypt[i,j]=(key+ASCII(F[i,j])+randno$^{(i+j)}$)mod256 (5)

Key→ the key generated from the keyword using the algorithm1.

ASCII(F(i,j))→ The Ascii value for the character at the **j**th position in the **i**th block.

Randno→ any random number chosen by the DO.

Mod 256-→ The result of the bigger value from the equation is reduced to 256 so that the size of the cipher text never increases.

randno$^{(i+j)}$)mod256→It is very difficult to find the values of i and j for a given plaintext-ciphertext pair. It is based on the concept of discrete logarithm problem[18].

Let **x, α, y** and **β** are non zero integers.

Suppose

$$\boldsymbol{\beta \equiv \alpha^x \bmod y} \qquad \textbf{(6)}$$

It is very difficult to find the value of x given **α, y** and **β.**

Even though there is a method called Pohlig-Hellman algorithm [18] to compute the discrete log, it is not possible to find the values of x, since it is changing for every character in the text.

5 Performance Analysis

We analyzed the performance in terms of storage, communication and computation complexity.

5.1 Storage Cost:

The storage cost of DO, TPA and CS(Cloud Server) are as follows.

Data Owner: The client needs to store only the security parameters like keyword1,keyword2, initial seed, randno, q(the number of characters in the block), the random function and secret key (Sk) constantly. So the storage cost of the Data Owner is O(1).

Server Side: The server has to store the complete file containing encrypted data (n bits) along with the meta data(n bits)[3]. So the storage cost at the server side is O(2n). It is illustrated in the Fig.7.

The metadata is generated for all the charaters in the plaintext depending on the position of charcters in the file and the secret key (**Sk**). Then the metadata is appended to the data file. So the size of the data file becomes doubled. Even though the data size is increased, the client needs to store only the random function and the secret key(**Sk).** The data file along with the metadata is stored in the cloud server.

Third Party Auditor: The TPA or the verifier has to store only the random function and secret key (Sk) constantly. So the storage cost of the TPA O(1).

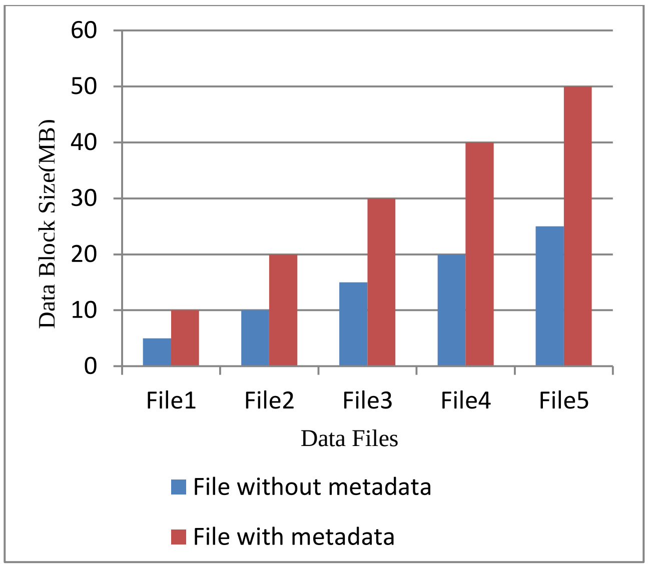


Fig.7: Comparison of file sizes with and without metadata

5.2 Computation Cost

We analyzed the computation cost of the DO,TPA and the cloud server.

Client: The client generates the key1,key2 and the binary sequence of length n blocks. The cost of this computation is O(n). The cost of encryption function is O(n). The cost of meta data generation is O(n). So total computation cost of the user is **O(n).**

Server: The computation done at the server is very less. It has to send the response message consiting of encrypted data and metadata for the challenge message. The computation cost is **O(1).**

TPA: The computation done at the verifier is minimum. It has to generate the challenge message, and verify the integrity by doing the inverse of random function. The computation cost of TPA is O(1).

5.3 Communication Cost

The communication cost between the client and server is O(n), between the verifier and the server is O(1) and between the client and TPA O(1). The storage and computation costs are summarized in the Table 2.

6 Results

A private cloud environment has been established with the open source eucalyptus cloud simulator tool. To install and configure an Ubuntu enterprise cloud , two Servers (Server1 and Server2) that run with 32-bit , 1GHz server version and two machines that run as a Desktop 32-bit version (Client and TPA) are required. The ubuntu desktop version is installed on Client and TPA so that the browsers can be used to access the web interface of UEC. The experiment was conducted using a setup consisting of two servers and one desktop.

Encryption and decryption algorithms are implemented in java and communication between client and server is implemented with the java remote method invocation concepts. The time taken for the encryption and decryption operations are given in the Table 3.

	[5]	[9]	[4]	[3]	[8]	[13]	[2]	**Proposed**
Confidentiality	No	Yes	Yes	No	No	No	No	Yes
Public Verifiability	Yes	Yes	Yes	No	No	No	Yes	Yes
Data Dynamics	Yes	Yes	Yes	No	No	No	Yes	Yes
Server Computation cost	O(log n)	O(log n)	O(ts)	O(1)	O(t)	O(t+s)	O(t logn)	O(1)
Verifier computation cost	O(log n)	O(log n)	O(ts)	O(1)	O(t)	O(t+s)	O(tlog n)	O(1)
Probability of corruption detection (insertion, deletion ,appending corruptions)	$1-(1-p)^t$	$1-(1-p)^t$	$1-(1-p)^{ts}$	1	$1-(1-p)^t$	$1-(1-p)^{ts}$	$1-(1-p)^{ts}$	1
Probability of corruption detection (data replacement)	$1-(1-p)^t$	$1-(1-p)^t$	$1-(1-p)^{ts}$	$1-(1-p)^{ts}$	$1-(1-p)^t$	$1-(1-p)^{ts}$	$1-(1-p)^{ts}$	$1-(1-p)^{ts}$

n- the number of blocks in the file, t- number of challenge requests to server ,s- number of sectors in a block
p-probability of corrupted blocks

Table 1: Comparison of our proposed scheme with other integrity checking protocols

Storage Cost			Computation Cost		
DO	TPA	Server	DO	TPA	Server
O(1)	O(1)	O(2n)	O(n)	O(1)	O(1)

Table 2:Storage and computation cost of our proposed scheme

We compared our scheme with the existing remote integrity checking methods. The comparison analysis of our proposed methods with the existing methods are illustrated in the Table 1. It shows that our scheme is the one which offers the highest probability of corruption detection. And also if there are corruptions like appending, deletion, and insertion of malicious data in the data stored in cloud , our scheme detects the corruptions with the highest probability of 1.

File Size (KB)	Encryption Time (milliseconds)	Decryption Time (milliseconds)
1	15	20
2	25	27
3	32	35
4	35	47
5	40	50
6	44	46
7	50	48
8	51	48
9	59	54
10	64	65

Table 3:Time of encryption and decryption of different files

Probability of detection of data replacement corruption is illustated in Fig.8. From this it is inferred that, the probability of detection of data replacement corruptions is higher in our proposed sytem and in the papers[2][3][4][8]. But for the other data corruptions like data deletion, data insertion and data appending, our proposed system has the highest probability detection of 1, that was not achieved through the previous integrity checking methods.

7 Conclusion

In this paper, we have analyzed various security challenges in cloud environment and proposed a suitable solution for confidentiality and integrity assurance by designing an efficient block cipher, 2-keys symmetric encryption technique. This proposed scheme can be applied for the secure storage of bulk data . After the detailed study, it is analyzed that it is the first method that detects the data corruptions with the probability of 1. This method is appropriate for the data that is dynamically changing. Our scheme requires less computation and communication cost ,and can be used for large-scale cloud storage systems. Our encryption scheme can also be applied for encrypting image and video files.

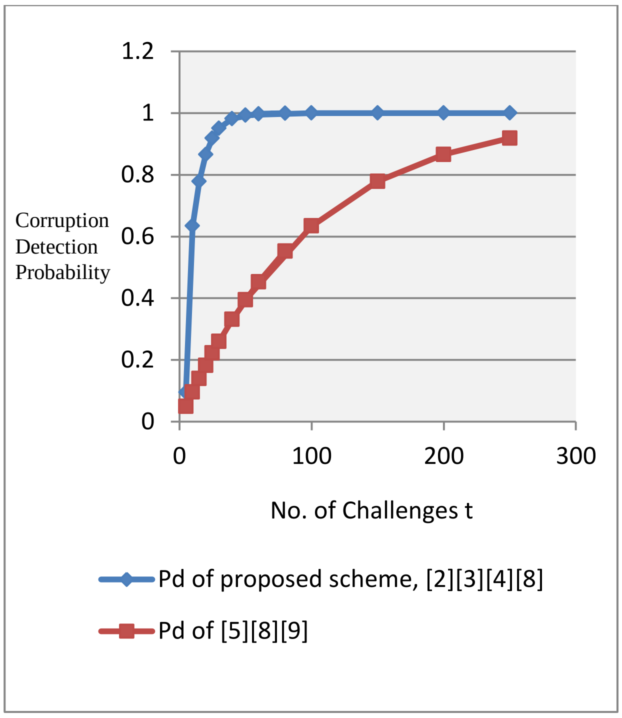


Fig.8: Probability of detection Pd of data replacement corruptions for 100 blocks, 10 sectors in a block and 1corrupted block

References

[1] Veeralakshmi Ponnuramu, and Latha Tamilselvan., (2014), ‘Encryption for Massive Storage in Cloud’ *in Computational intelligence in Data Mining, Volume 2,pp 27-38, Smart Innovation, Systems ,and Technologies(Springer), volume 32.*

[2] C.Wang , Q.Wang, S.S.M.Chow, K.Ren,W.Lou, (2013) , ‘Privacy-Preserving Public Auditing for Secure Cloud Storage’ , *IEEE Transactions on Computers* , Vol 62, No.2,.

[3] Ponnuramu Veeralakshmi. and Latha Tamilselvan, (2012).’ Data integrity proof and secure computation in cloud computing’. *J. Comput. Sci., 8: 1987-1995.*

[4] Kan Yang, Xiaohua (2013) ‘ An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing’, *IEEE Transactions On Parallel and Distributed systems*,VOL 24, NO. pp.1717-1726.

[5] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, (2012),’Toward Secure and Dependable Storage Services in Cloud Computing’, *IEEE Transactions On Services Computing*,VOL. 5, NO. 2, pp.220-232.

[6] Pearson,(2009), ‘Taking Account of Privacy when Designing Cloud Computing Services’, in *Proceedings of ICSE-Cloud’09*, Vancouver.

[7] A.Juels and B. S. Kaliski, Jr.,(2007),’Pors: proofs of retrievability for large files,’ in CCS .’07: *Proceedings of the 14th ACM conference on Computer and communications security*. New York, NY, USA: ACM, pp. 584–597.

[8] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song,(2007), ‘Provable data possession at untrusted stores,’ in CCS ’07*: Proceedings of the 14th ACM conference on Computer and communications security. New York*, NY, USA: ACM, pp. 598– 609.

[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou,(2011), ‘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.

[10] M. Jensen, et al.,(2009), ‘On Technical Security Issues in Cloud Computing,’ in *IEEE International Conference on Cloud Computing*, Bangalore, India, pp. 109-116.

[11] Lifei Wei , Haojin Zhu, Zhenfu Cao, Weiwei Jia,(2010), ‘SecCloud: Bridging secure storage and computation in cloud’, in ICDCS’10.

[12] C.Wang, Q. Wang, K. Ren, and W. Lou, (2009), ’Ensuring Data Storage Security in Cloud Computing,’ *in Proc. of IWQoS’09*.

[13] H. Shacham and B. Waters,(2008), ‘Compact Proofs of Retrievability,’, *Proc. 14th Int’l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology* pp.90-107.

[14] E.-C. Chang and J. Xu,(2008), ‘Remote integrity check with dishonest storage server,’ *in Proc. Of ESORICS’08.Berlin, Heidelberg: Springer-Verlag*, pp. 223–237.

[15] C. Wang, Q. Wang, K. Ren, and W. Lou,(2010), ‘Privacy- Preserving Public Auditing for Data Storage Security in Cloud Computing,’*Proc. IEEE INFOCOM*, pp. 525-533.

[16] Y. Zhu, H. Hu, G. Ahn, and M. Yu,(2012), ‘Cooperative Provable Data Possession for Integrity Verification in Multi-Cloud Storage’, *IEEE Trans. Parallel and Distributed Systems, vol. 23, no. 12, pp. 2231-2244.*

[17] Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S.S. Yau, (2007), ’Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds,’,*Proc. ACM Symp. Applied Computing*, pp. 1550-1557.

[18] W. Stallings, Cryptography and network security principles and practice, Fourth edition, Prentice hall, 2007