Secured Storage for Dynamic Data in Cloud

Veeralakshmi Ponnuramu, Department of Computer Science and Engineering
B.S.Abdur Rahman University, Chennai, India
E-mail: veerphd1@gmail.com

Dr. Latha Tamilselvan, Department of Information Technology
B.S.Abdur Rahman University, Chennai, India
E-mail: latha_tamilselvan@yahoo.com

Keywords: confidentiality, remote integrity verification, dynamic data operations, public verifiability, symmetric encryption, data storage security, privacy preserving auditing

Received: December 21, 2015

Cloud is a growing computing paradigm in which the services and resources are provisioned dynamically through internet. In cloud, the users' data can be stored in the remotely located data servers that are maintained by cloud service providers to relieve the users from the local storage and maintenance. The major security challenge with cloud computing is that the users cannot have direct control over the remotely stored data. The imperative security concerns in cloud are the integrity and confidentiality of data. Many existing remote integrity checking methods fail to serve for the data that can be dynamically updated. 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 (TPA) for verifying the data integrity. It is ensured that the data stored in the untrusted cloud server is confidential and consistent by using a data encryption algorithm called 2-Keys Symmetric Encryption. Unlike other encryption algorithms, this encryption algorithm needs lesser computation overhead. Encryption and decryption algorithms are developed in java and Remote Method Invocation (RMI) concepts are used for communication between client and server. Simulation environment is set up with the eucalyptus tool. The performance analysis and simulation results prove that our proposed scheme is secure and proficient and it reduces the computation cost of server and verifier.

Povzetek: Ta članek predlaga postopek za zagotavljanje varnosti za dinamično spreminjanje podatkov, ki so shranjeni v oblaku.

1 Introduction

Cloud is an on-demand, pay-by-use model for sharing a pool of computing resources like servers, CPU cycles, memory, applications, storage and services that is managed by cloud service providers. The services can be easily provisioned from the cloud providers and released with minimum endeavor by the cloud users. The users can access data and applications from remote servers with the fixed or mobile devices. Cloud storage becomes an increasing attraction in cloud computing paradigm. Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Services (S3) are the well-known examples of cloud storage services. With the cloud, the small organizations can hire the resources from the cloud rather than purchasing them and avoid the capital costs for software and hardware. With cloud, IT infrastructure can be easily adjusted to accommodate the changes in demand [5].

There are various [6], [10] issues like security, scalability, availability, resource scheduling, data migration, memory management, data privacy, data management, reliability, load balancing, access control in cloud because it uses many technologies including virtualization, networks, transaction management, databases and operating systems. Cloud moves the applications, software and databases to the large data centers that are located anywhere in the world, where the servers can't be trustworthy. This unique feature of cloud imparts many new security challenges like confidentiality and integrity. At the same time, the cloud offers many benefits like enhanced collaboration, limitless flexibility, portability and simpler devices. To enjoy the 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. So the existing methods like SHA, MD5 [16] can't be directly used. The simple 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 effective methods are required for assuring the

confidentiality and integrity of data stored in the cloud with the minimum overhead.

Recently many remote integrity checking methods [13], [8], [14], [1], [6], [15] 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 persons in authorized ways. Updates of data file include writing, appending to the existing data, changing, deleting the data and creation of new information.

There are two kinds of encryption algorithms. That is symmetric and asymmetric encryptions. In symmetric algorithms, the same key can be used for both encryption and decryption. Symmetric algorithms are highly secured and can be executed in high speed. In case of asymmetric algorithms, different keys are used for encryption and decryption [16]. In this, data can be encrypted using a public key, and decrypted with a private key. Asymmetric encryption algorithms (called as public-key algorithms) need a key of 3,000 bits to produce the same level of security as that of a symmetric algorithm with a 128-bit key. Since the asymmetric encryption algorithms are slow, they cannot be used for encrypting bulk of data. So, in this paper, a novel symmetric encryption algorithm has been proposed for maintaining confidentiality and assuring data integrity.

2 Related work

2.1 Security issues in cloud

There are numerous security issues in cloud as the customers are not having direct control over the stored data in cloud. Jensen et al., [11], [12] discussed 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 browser security, WS-security, cloud integrity, transport layer security, and binding issues in the field of cloud.

2.2 Merkle hash tree (MHT)

Wang et al., [3] verified the correctness of data stored in server by allowing the Third Party Auditor. With the aid of Merkle hash tree it is possible for the clients to perform block-level operations on the data files by preserving the level of data correctness assurance. In this scenario, chances are there for the third party auditor to misuse the data while doing the verification operation. Lifei Wei et al., [9] established a new mechanism to verify the correctness of computations (addition, subtraction, multiplication, division, etc.) done by the cloud provider. For that, they have used the Merkle hash tree for checking the computation correctness. The only criteria are the number of computations submitted to the server must be in the power of 2, since the Merkle hash tree has the 2n number of leaves.

2.3 Advance computation of tokens

A storage correctness model for verifying the correctness of stored data by calculating a few numbers of tokens was proposed by Wang et al., [3]. This insists the user to pre-compute a number of verification precomputed tokens, each of them covering a random subset of data blocks. It allows the cloud user to challenge the cloud server with a set of pre-computed tokens. Once accepting the challenge token, the cloud server computes a signature over the specified data blocks and returns the signature to the cloud user. The signatures returned by the provider should match the relevant 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 Proof of retrievability scheme (POR)

For verifying the data integrity some sentinel characters were embedded in the data file by A.Juels and B. S. Kaliski [1]. These sentinels were hidden in the data blocks. In the verification phase, the user can challenge the server by mentioning the positions of sentinels and request the provider to return the relevant sentinel values. This procedure allows the user to challenge the server for a limited number of times by knowing the positions of the sentinel values in advance. G.Ateniese et al., [6] proposed a new model called “Provable Data Possession” to ensure the possession of files stored on the untrusted server. They had used RSA- based homomorphic tags for assessing outsourced data. Here also the user needs to pre-compute the tags and store 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 [7]. For ensuring the remote integrity, the MAC and reedsolomon code were used [5].

2.5 Dynamic data operations

Many of the existing remote checking methods support only static data [13], [1], [6], [7], [5]. These are not featured with the methods for handling dynamically changing data. Several methods have been proposed for provisioning dynamic data in cloud [4], [15], [17], [8]. Among these, some of the papers are not offering the support for block insertion operations and are detecting the data corruption with a lesser probability [8]. For the high probability of detection of data corruption, it is needed for the increased number of challenges to the server from the client or TPA. These methods are not considering the issue of confidentiality. In this paper, a new method for assuring confidentiality and integrity of dynamically changing data is proposed.

Our scheme uses a stream cipher encryption algorithm called 2-Keys Symmetric Encryption [18] 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 [13]. 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 and appending are conceivable without retrieving the entire data from server by using the linked list data structure. Here public auditability 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 depend on the CSPs for data computation. The client can be the individual user or an organization.

Cloud Service Providers: It can be the organization 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.

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

2. Public Auditability: It allows anyone including TPA to verify the integrity 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 are 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 Figure 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.

(1). Key generation

The DO generates two symmetric keys by using the algorithm1. Two keywords (keyword1 and keyword2) of variable length are given as the input from the user for generating the keys. From the keywords two keys (key1 and key2) are generated by making use of the ASCII values of characters in the keywords and their position in the keyword. The two keywords are kept secret by the DO by storing it in the key store of data owner.

(2). 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.

$$Xn+m = (C0Xn+C1Xn+1+C2Xn+2+\ldots Cm-1Xn+m-1) \bmod 2 \quad (1)$$

For generating the recurrence relation, the user needs an initial seed value m, the initial vector values like (X1, X2, X3, X4., Xm) and the coefficient values like (C0,C1,C2 C3.,Cm-1).For example, for the initial seed m=5, initial vector (0,1,0,0,0) i.e. X1=0; X2=1; X3=0; X4=0; X5=0 and for the coefficient (1,0,1,0,0) i.e. C1=1; C2=0; C3=1; C4=0; C5=0, m=5 the recurrence relation is like the equation 2.

$$Xn+5=Xn+Xn+2 \quad (2)$$

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

(3). Encryption

To ensure the confidentiality of data, the DO encrypts each data block using the 2-keys Symmetric Encryption algorithm [18]. It takes as input the data file F, binary sequence, key1,key2 and a secret random number for encryption and produces the cipher text in file F'. This

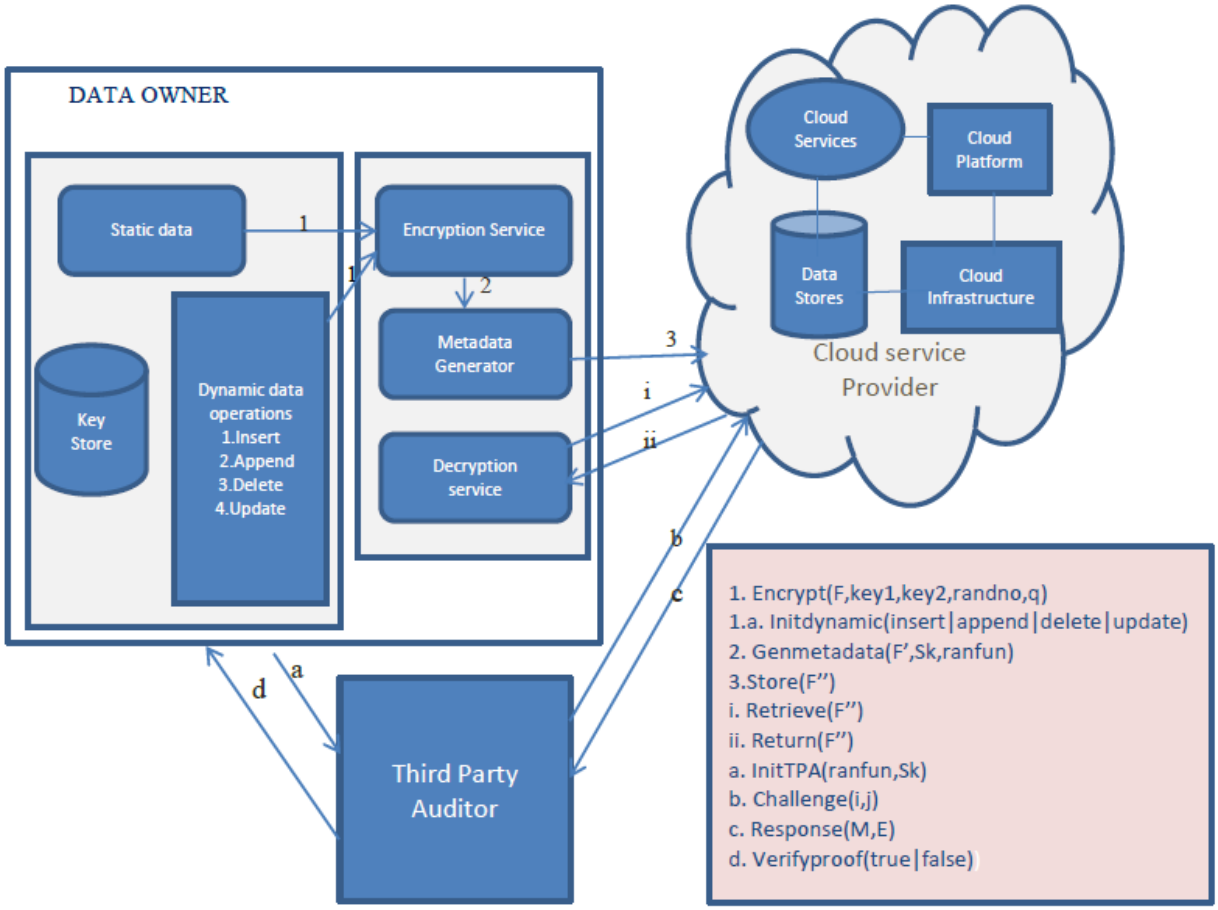

Figure 1: System Architecture Diagram.

encryption technique splits the data file **F** into **n** data blocks **db1, db2, db3., dbn** considering that each of the n data blocks contains s bytes.

(4). Metadata generation

After encrypting the data, the DO computes the metadata of the encrypted data blocks to ensure the integrity of data stored in the server. To generate the metadata the algorithm proposed in [13] can be used.

It is using some random functions involving the position of characters in the file **F'** and a secret key (**Sk**) chosen by the DO. The DO has to keep the random function as a secret one. The DO can issue the random function and the secret key (**Sk**) to the TPA for verifying the integrity of data.

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

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

The generated metadata is concatenated with the encrypted data of each block and is stored in the F''.

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} consisting 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 and sends it to the untrusted server. Upon receiving the challenge message from the TPA, the server generates response message and send it to the TPA. Using this, the TPA checks the integrity of the message as discussed with [13].

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 [8], Merkle Hash Tree [14]. Here the simple data structure called the linked list is used. 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) do not know 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 Figure 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.

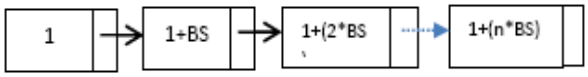

Figure 2: Linked List Construction.

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 block at the middle, append the block at the end and insert a node at the position **k** in the linked list maintained with the DO. This scheme performs the insertion operation without re-computing the metadata and encrypted data for all blocks that have been shifted after inserting a block. It calculates the encrypted metadata only for the block which we are going to insert. Block insertion can be done using the block insertion algorithm.

Block Insertion Algorithm

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. (i.e.) **db*** and the position **k** after which it is to be inserted.
2. Perform the encryption for the new data block **db*** using the 2-keys symmetric encryption algorithm as explained in [18] 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*,}.

The DO updates the linked list as in the Figure 3.

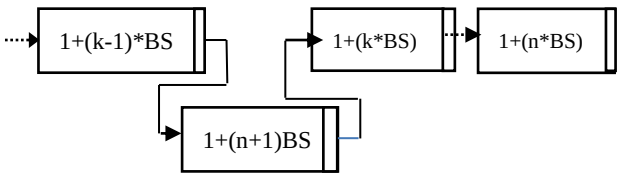

Figure 3: Linked list after Block insertion.

3.3.3.2 Block modification

Block modification can be simply done using our scheme. If the user wants to modify a block db2 at the position k with dbm, 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, make 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

Appending a block is also very simple in our method. To append a block dba at the end, the data owner computes the encrypted data and metadata without disturbing the other blocks. Then the user makes an append request **Append (F'', dba)** to the cloud server to append the data block dba at the end of the File **F''**.

The server appends the data block **dba** at the end as **{db1, db2, db3., dbn, dba}. T**he client updates the linked list as in the Figure 4.

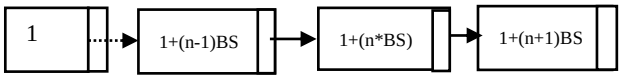

Figure 4: Linked list after Block Appending.

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 update the file as **{db1, Φ, db3., dbn**}. The client updates the node at the 2nd position with **Φ**.

3.3.4 Decryption phase

The user uses the secret key (Sk) and the reverse of the random function used to generate the metadata and recovers the encrypted data. From the encrypted data, the DO gets the original data using the decryption algorithm [18]. The DO generates the binary sequence from the recurrence relation, initial vector and the coefficients and the keys key1, and key2 from the keywords keyword1, keyword2 respectively for decryption. The DO divides the encrypted data file encdata.txt into **n** data blocks **db1, db2, db3., dbn** considering each n data blocks contains **s** bytes **like b1, b2, b3., bs.**

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, there is a Third Party Auditor (TPA), 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.

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

Support Dynamic Operations: After storing the data in server, the user can update the data dynamically. It should dynamic operations like block insertion, block modification, block deletion and block appending. Our system supports efficiently supports all the dynamic operations that is not discussed with [13].

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 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. For data deletion, data insertion and data appending corruptions, our system detects the corruption 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 (Pd) of data replacement corruption, consider the following assumptions.

- The file **F** contains the **n** blocks and the probability to pick any block 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.

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

The Figure 5 illustrates the probability detection of data corruptions like data replacements, data appending, data insertion and data deletion for the file containing n=1000 blocks, s=20 sectors in a block and 1 corrupted block. This figure infers that the data appending, data insertion and data deletion corruptions are identified by the highest probability of 1 requiring minimum number of challenges. The data replacement corruptions are identified by somewhat high probability. In this the number of challenges is proportional to the number of corrupted blocks.

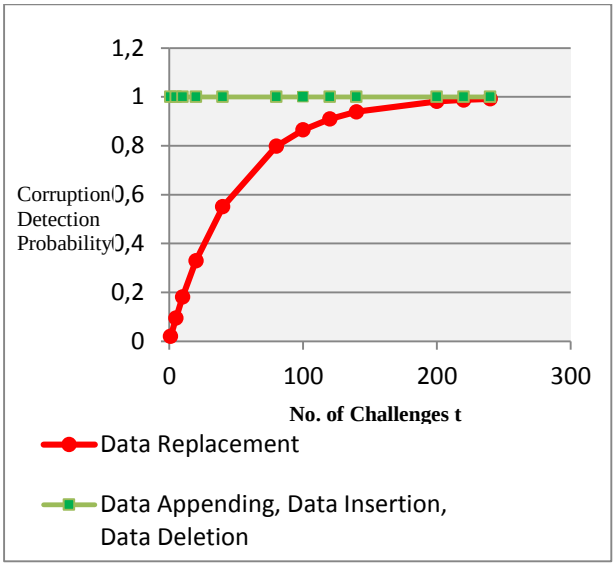

Figure 5: Probability of detection Pd of data corruptions for 1000 blocks, 20 sectors in a block and, 1corrupted block.

4.2 Confidentiality

We analyzed the confidentiality of our scheme. Here we have designed a symmetric stream cipher encryption algorithm. Stream ciphers are suitable for the larger data than the block cipher methods. In block cipher, the data is split into blocks. The same encryption algorithm and key are used for encrypting all the blocks in the data. If a same block is repeated again in the plaintext, in electronic code book (ECB) mode, we get the same cipher text for the repeated blocks. The modes of operations like Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB) are not suitable for dynamic data. So, it is decided that block ciphers are not suitable for larger dynamically changing data.

The existing stream ciphers 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 to generate the key, it is required to use the LFSR sequence, blub-blub generator, or recurrence relations for generating binary sequence. This binary sequence is prone to correlation attack. So we have used 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. (X1, X2, X3, X4., Xm) initial coefficients (C0, C1, C2, C3., Cm-1) random number randno, number of characters in the block –q are all the secret information maintained by the cloud user.

So the brute force attack is not conceivable in this 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 frequencies. Figure 6 shows the frequency of characters of the encrypted data.

Correlation attacks:

The attacks try to extract some information about the initial state from the output stream. Here, since the recurrence relation 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 hacker 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

$$\text{Encrypt}[i,j] = (\text{key} + \text{ASCII}(F[i,j]) + \text{randno}(i+j)) \bmod 256 \quad (5)$$

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.

Mod256→the result of the bigger value from the equation 5 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-cipher text pair. It is based on the concept of discrete logarithm problem [16].

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

Suppose

$$\beta \equiv \alpha x \bmod y \quad (6)$$

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 [16] to compute the discrete log it is not possible in this algorithm to find the values of x, since it is changing for every character in the text.

5 Performance analysis

The performance in terms of storage, communication and computation complexity is analyzed.

5.1 Storage cost:

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

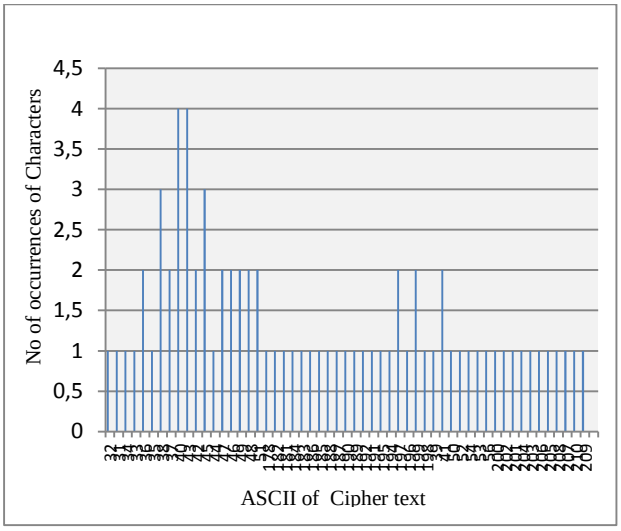

Figure 6: Frequency of characters of encrypted data.

Data Owner: 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) [13]. So the storage cost at the server side is O(2n). It is illustrated in the Figure 7.

The metadata is generated for all the characters in the plaintext depending on the position of characters 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).

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 metadata generation is O(n). So total computation cost at the user is O(n).

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

TPA: The computation done at the verifier is lesser. 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).

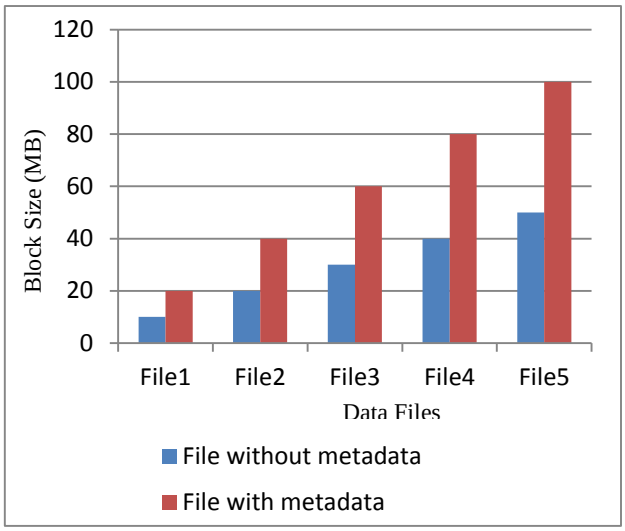

Figure 7: Comparison of file sizes with and without metadata.

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 1.

Table 1: Storage and computation cost of our proposed scheme.

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

6 Results

.A private cloud environment has been established with the open source eucalyptus cloud simulator. To install and configure an Ubuntu enterprise cloud, two Servers (Server1 and Server2) that run with 32-bits, 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 (RMI) concepts. We compared our scheme with the existing remote integrity checking methods. The comparison analysis of our proposed method with the existing methods is illustrated in the Table 2. 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.

Probability of detection of data replacement corruption is illustrated in Figure 8. From this it is inferred that, the probability of detection of data replacement corruptions is higher in our proposed system and in the papers [8], [13], [5], [2]. 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, various security challenges in cloud environment have been analyzed and an appropriate solution for providing confidentiality and ensuring integrity of dynamically changing data has been proposed by using an efficient block cipher, 2-Keys Symmetric Encryption technique with the linked list data structure. This scheme can also 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. Since this system requires less computation and communication cost, it can be used for large-scale cloud storage systems. This can be further extended by finding a method that applies this mechanism for the security of unstructured data.

Table 2: Comparison of our proposed scheme with other integrity checking protocols.

Issues	**[13]**	**[14]**	**[8]**	**[13]**	**[6]**	**[7]**	**[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(t logn)	O(1)
Probability of corruption detection (insertion, deletion, appending)	$1-(1-p)^t$	$1-(1-p)^t$	$1-(1-p)^{ts}$	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

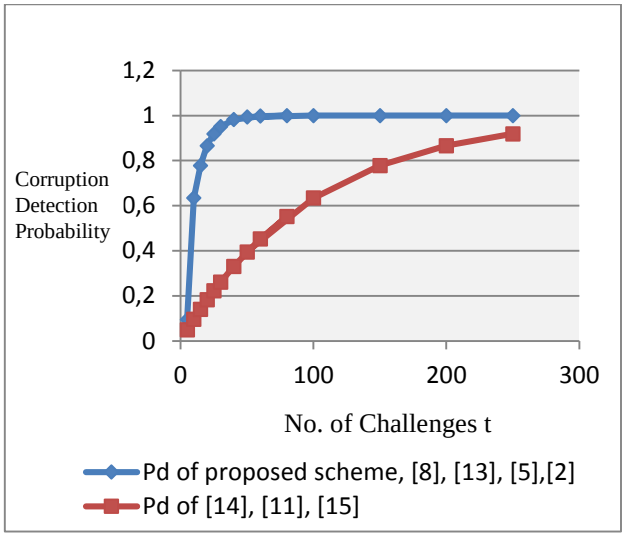

Figure 8: Probability of detection Pd of data replacement corruptions for 100 blocks, 10 sectors in a block and 1corrupted block.

References

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

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

[3] C. Wang., Q. Wang., K. Ren., and W. Lou, (2009), "Ensuring Data Storage Security in Cloud Computing," in Proc. Of IWQoS' 09.

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

[5] 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.

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

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

[8] Kan Yang, and 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. 9, pp.1717-1726.

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

[10] M. Armbrust et al., (2009), "Above the clouds: A berkeley view of cloud computing", ECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28,.

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

[12] Pearson (2009), "Taking Account of Privacy when Designing Cloud Computing Services", in Proceedings of ICSE-Cloud'09, Vancouver.

[13] Ponnuramu Veeralakshmi, and Latha Tamilselvan, (2012), "Data Integrity Proof and Secure Computation in Cloud Computing'. J. Comput.Sci., 8: 1987-1995.

[14] 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.

[15] 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.

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

[17] 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.

[18] 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.