International Research Journal of Mathematics, Engineering and IT
Vol. 4, Issue 6, June 2017 **Impact Factor- 5.489**
ISSN: (2349-0322)
© Associated Asia Research Foundation (AARF)
Website: www.aarf.asia**Email** : editor@aarf.asia , editoraarf@gmail.com

STORAGE IN CLOUD COMPUTING BY SECURE ERASURE CODE

Rinu Sam S

Sivaji College of Engineering and Technology, Manivila, Tamil Nadu, India.

ABSTRACT

A cloud storage system consisting of a collection of storage servers provides long term services over the Internet. Storing data in a third party's cloud causes serious concern over data confidentiality. General encryption scheme has some limitations. Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. We propose a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure and robust data storage and retrieval, but also lets a user forward his data in the storage servers to another user without retrieving the data back. The main technical contribution is that the proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages. Our method fully integrates encrypting, encoding, and forwarding. We analyze and suggest suitable parameters for the number of copies of a message dispatched to storage servers and the number of storage servers queried by a key server. These parameters allow more flexible adjustment between the number of storage servers and robustness

Keywords: Decentralized erasure code, proxy re-encryption, secure storage system

1. Introduction

Cloud Computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access (fig.1). This technology allows for much more efficient computing by centralizing data storage, processing and bandwidth. That is Cloud Computing is the method of delivering resources (hardware or software resources) as a service over the internet. And Cloud is a collection of servers that is capable of providing different services to users [1-3]. A simple example of cloud computing is Yahoo email, Gmail, or Hotmail etc. All you need is just an internet connection and you can start sending emails. The server and email management software is all on the cloud (internet) and is totally managed by the cloud service provider Yahoo, Google etc. The consumer gets to use the software alone and enjoy the benefits.

1.1. Cloud Storage

There are many types of services provided by the cloud. They are

- Infrastructure as a service (IaaS)
- Platform as a service (PaaS)
- Software as a service (SaaS)
- Storage as a service (STaaS)
- Security as a service (SECaaS)
- Data as a service (DaaS)
- Database as a service (DBaaS)
- Test environment as a service (TEaaS)
- Desktop virtualization

Cloud Providers who provide Storage as a Service provides online storage of data in the database of the cloud. These hosting companies operate large data centers, and people who require their data to be hosted buy or lease storage capacity from them. The datacenter operators, in background, virtualizes the resources according to the requirements of the customer and expose them as storage pools, which the customers can themselves use to store files or data objects. Physically, the resource may span across multiple servers. Cloud computing offers many benefits, but it also is vulnerable to threats. As the uses of cloud computing increase, it is highly likely that more criminals will try to find new ways to exploit vulnerabilities in the system [4]. There are many underlying challenges and risks in cloud computing that increase the threat of data being compromised [5]. To help mitigate the threat, cloud computing stakeholders should invest heavily in risk assessment to ensure that the system encrypts to protect data; establishes trusted foundation to secure the platform and infrastructure; and builds higher assurance into auditing to strengthen compliance. Security concerns must be addressed in order to establish trust in cloud computing technology.

Storing data in a third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a cryptographic method before applying an erasure code method to encode and store messages [6]. When he wants to use a message, he needs to retrieve the codeword symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. There are three problems in the above straightforward integration of encryption and encoding. First, the user has to do most computation and the communication traffic between the user and storage servers is high. Second, the user has to manage his cryptographic keys. If the user's device of storing the keys is lost or compromised, the security is broken. Finally, besides data storing and retrieving, it is hard for storage servers to directly support other functions. For example, storage servers cannot directly forward a user's messages to another one. The owner of messages has to retrieve, decode, decrypt and then forward them to another user.

In this paper, we address the problem of forwarding data to another user by storage servers directly under the command of the data owner. We consider the system model that consists of distributed storage servers and key servers (fig.2). Since storing cryptographic keys in a single device is risky, a user distributes his cryptographic key to key servers that shall perform cryptographic functions on behalf of the user. These key servers are highly protected by security mechan-isms. To well fit the distributed structure of systems, we require that servers independently perform all operations. With this consideration, we propose a new threshold proxy re-encryption scheme and integrate it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding opera-tions over encrypted messages and forwarding operations over encrypted and encoded messages. The tight integra-tion of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of a distributed structure is challenging. Our system meets the requirements that storage servers independently perform encoding and re-encryption and key servers independently perform partial decryption. Moreover, we consider the system in a more general setting than previous works. This setting allows more flexible adjustment between the number of storage servers and robustness.

1.2. Erasure code for storage applications

In this work it deals with storage systems, which includes researchers, programmers, and managers of disk array systems, distributed storage systems, wide-area storage systems, and peer-to-peer storage systems. No hard-core math or programming knowledge is required. The goal is for the participants to come away with a basic understanding of how erasure codes may be used in storage systems, including all the available techniques and technologies, plus their performance tradeoffs and their complexity [7]. Erasure Code is a FEC (forward error correction) which transforms a message of k symbols into a longer message with n symbols or transforms a message of k symbols into n blocks of size m and original message can be recovered from a subset of n symbols.

2. Background and Related Work

We briefly review distributed storage systems, proxy re-encryption schemes, and integrity checking mechanisms.

2.1. Distributed Storage Systems

At the early years, the Network-Attached Storage (NAS) and the Network File System (NFS) provide extra storage devices over the network such that a user can access the storage devices via network connection. Afterward, many improvements on scalability, robustness, efficiency, and security were proposed.

A decentralized architecture for storage systems offers good scalability, because a storage server can join or leave without control of a central authority. To provide robust-ness against server failures, a simple method is to make replicas of each message and store them in different servers. However, this method is expensive as z replicas result in z times of expansion.

One way to reduce the expansion rate is to use erasure codes to encode messages. A message is encoded as a codeword, which is a vector of symbols, and each storage server stores a codeword symbol. A storage server failure is modeled as an erasure error of the stored codeword symbol. Random linear codes support distributed encoding, that is, each codeword symbol is independently computed. To store a message of k blocks, each storage server linearly combines the blocks with randomly chosen coeffi-cients and stores the codeword symbol and coefficients. To retrieve the message, a user queries k storage servers for the stored codeword symbols and coefficients and solves the linear system.

We addressed robustness and confidentiality issues by presenting a secure decentralized erasure code for the networked storage system. In addition to storage servers, their system consists of key servers, which hold cryptographic key shares and work in a distributed way. In their system, stored messages are encrypted and then encoded. To retrieve a message, key servers query storage servers for the user. As long as the number of available key servers is over a threshold t, the message can be successfully retrieved with an overwhelming probability.

2.2. Proxy Re-Encryption Schemes

Proxy re-encryption schemes are proposed by Mambo and Okamoto In a proxy re-encryption scheme, a proxy server can transfer a ciphertext under a public key PK_A to a new one under another public key PK_B by using the re-encryption key $RK_{A!B}$. The server does not know the plaintext during transformation. Ateniese et al. proposed some proxy re-encryption schemes and applied them to the sharing function of secure storage systems. In their work, messages are first encrypted by the owner and then stored in a storage server. When a user wants to share his messages, he sends a re-encryption key to the storage server. The storage server re-encrypts the encrypted messages for the authorized user. Thus, their system has data confidentiality and supports the data forwarding function. Our work further integrates encryption, re-encryption, and encoding such that storage robustness is strengthened.

Type-based proxy re-encryption schemes proposed by Tang provide a better granularity on the granted right of a re-encryption key. A user can decide which type of messages and with whom he wants to share in this kind of proxy re-encryption schemes. Key-private proxy re-encryption schemes are proposed by Ateniese et al. In a key-private proxy re-encryption scheme, given a re-encryption key, a proxy server cannot determine the identity of the recipient. This kind of proxy re-encryption schemes provides higher privacy guarantee against proxy servers. Although most proxy re-encryption schemes use pairing operations, there exist proxy re-encryption schemes without pairing.

2.3 Integrity Checking Functionality

Another important functionality about cloud storage is the function of integrity checking. After a user stores data into the storage system, he no longer possesses the data at hand. The user may want to check whether the data are properly stored in storage servers. The concept of provable data possession and the notion of proof of storage are proposed. Nevertheless all of them consider the messages in the clear text form.

3. System Architecture Design

The system Architecture defines the operational flow of the entire system (fig.3). In this paper, we first present our Storage structure in cloud, which uses the secure code, based scheme and distributed storage for effective storage mechanism.

3.1 Distributed Storage in Cloud

Cloud storage system has a distributed storage mechanism (fig.4). The message of a user A can be split and then encrypted and stored in multiple server. In normal case only encryption is done. Erasure code base mechanism provides distributed storage of the message in the cloud system making the data of the user more secure. Similarly it can be retrieved by merging data distributed across the cloud.

Here data is encrypted twice for more security. After first encryption data is split according to the choice of the data owner. Then each of the split block is re-encrypted and zipped. Then again it is the data owner who decides where he wants to store each block. Then storing of the data blocks can be done in a distributed fashion among all the servers of the cloud. Anyone who is authorized to get the data can be retrieved from these distributed storage system with appropriate secret key and thus data is forwarded from the distributed storage system to the requested user.

4. Modules

This project has mainly divided in to six major modules. Each module has well designed to accomplish different tasks, those are listed and designed as part of system development and the different authorization based on the designated level. Each module has different sub-modules too to make the system a power full one. All the major tasks makes an effective approach for secure storage of data in the cloud server in the cloud computing environment.

The different modules available in this system are

- Process Encryption
- File Rifting
- Proxy Re-encryption
- Data Storage on Cloud
- Data Retrieval from cloud
- Data Forwarding

4.1. Process Encryption

In this module, we have to define the process we use for encryption. The process can be encrypted by using cryptographic keys. Since client files are stored in the cloud server, they have lesser security options. To overcome these securities we have implemented the crypto process.

Each user A is assigned a public-secret key pair (PKA, SKA). User A distributes his secret key SKA to key servers such that each key server KSi holds a key share SKA,i, $1 \leq i \leq m$. The key is shared with a threshold t.

4.2. File Rifting

To achieve an effective data exchange for cloud computing the prevention of file from intruders takes a vital role. After the process can be encrypted, it can be Spitted as different process. Splitting process is decided by the data owner. He decides the number of output files after splitting.

The fragmentation makes a bit higher complexity to gain complete information of a file. Thus an intruder does not get any idea by getting only any of the output file. File rifting process strengthen the process of sending and storing data in the cloud.

4.3. Proxy Re-encryption

Each of the split output is again encrypted separately. All the k subparts of the message m is encrypted again.

User A RE encrypts his message M and dispatches it to storage servers. A message M is decomposed into k blocks m1,m2, . . . , mk and has an identifier ID. User A encrypts each block mi into a cipher text Ci and sends it to v randomly chosen storage servers.

4.4. Data Storage On Cloud

The re-encrypted k message blocks can be saved anywhere on the cloud. The cloud consists of n number of servers and it is the data owner /client who decides where his re-encrypted message blocks be saved.

A single message block or a group of message blocks can be stored in a server in the cloud. The data stored in the cloud is very secure as it is split and saved in different locations of the cloud.

4.5. Data Retrieval

This is the process of retrieving data which is stored in the cloud. When the user needs back the file which is in cloud server the user can retrieve the re-encrypted files from multiple servers.

It consists of the following processes.

- Decrypt the files individually by referring the keys in key servers
- Decrypted individual files are then merged to get the whole file
- Again decrypted to get the original file.

4.6. Data Forwarding

Data forwarding lets a user forward his data in the storage servers to another user without retrieving the data back. User A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key .Data forwarding process forwards the data to the requested user.

5. Algorithm

5.1. DES Algorithm for Encryption

It operates on blocks of 64 bits using a secret key that is 56 bits long (fig.5). The original proposal used a secret key that was 64 bits long. It is widely believed that the removal of these 8 bits from the key was done to make it possible for U.S. government agencies to secretly crack messages

DES is a block cipher--meaning it operates on plaintext blocks of a given size (64-bits) and returns cipher text blocks of the same size. Thus DES results in a permutation among the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32 bits each, a left half block L and a right half R. (This division is only used in certain operations.)

The DES algorithm uses the following steps:

Step 1: Create 16 sub keys, each of which is 48-bits long.

Step 2: Encode each 64-bit block of data.

6 Conclusion & Future Work

We integrate a newly proposed threshold proxy re-encryption scheme and erasure codes over exponents. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of k blocks that are encrypted and encoded to n codeword symbols, each key server only has to partially decrypt to codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure.

6.1. Future Works

- Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. So maintain a Central authority for all the cloud storage activities.
- General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because few operations are supported over encrypted data. So we can use an effective encryption algorithm based on El-Gamal & MD5.

- Checking the correctness of the data after retrieving from the cloud database.

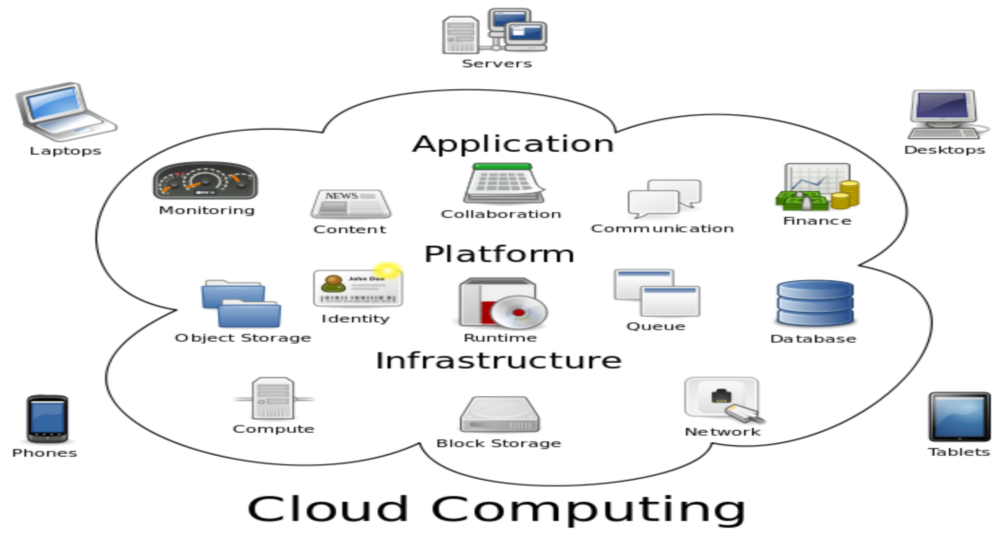

Fig 1: Cloud Computing

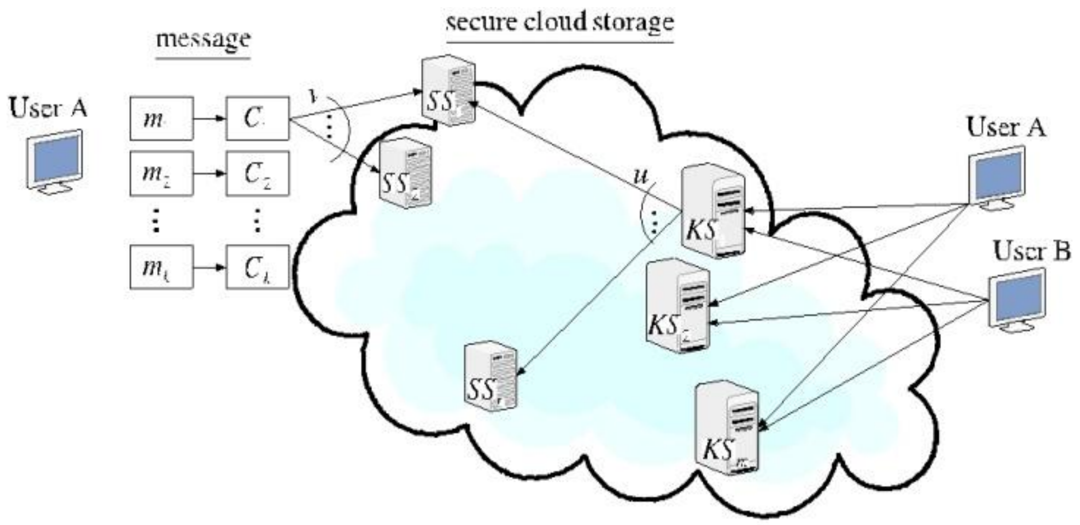

Fig: 2.Cloud Storage

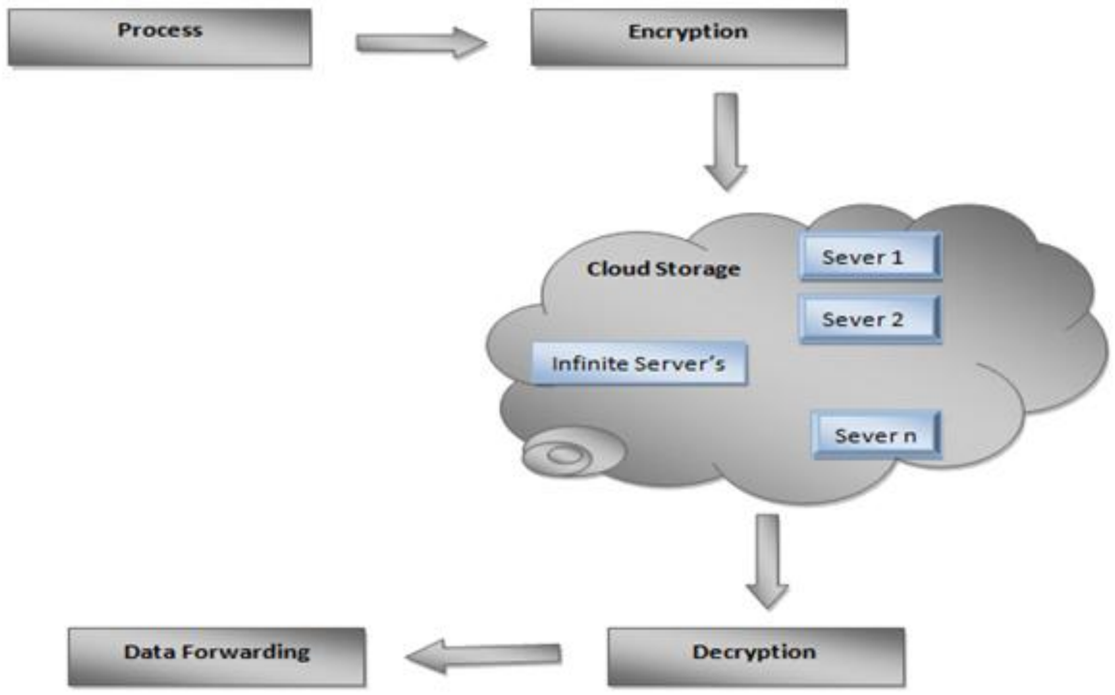

Fig: 3. System Architecture

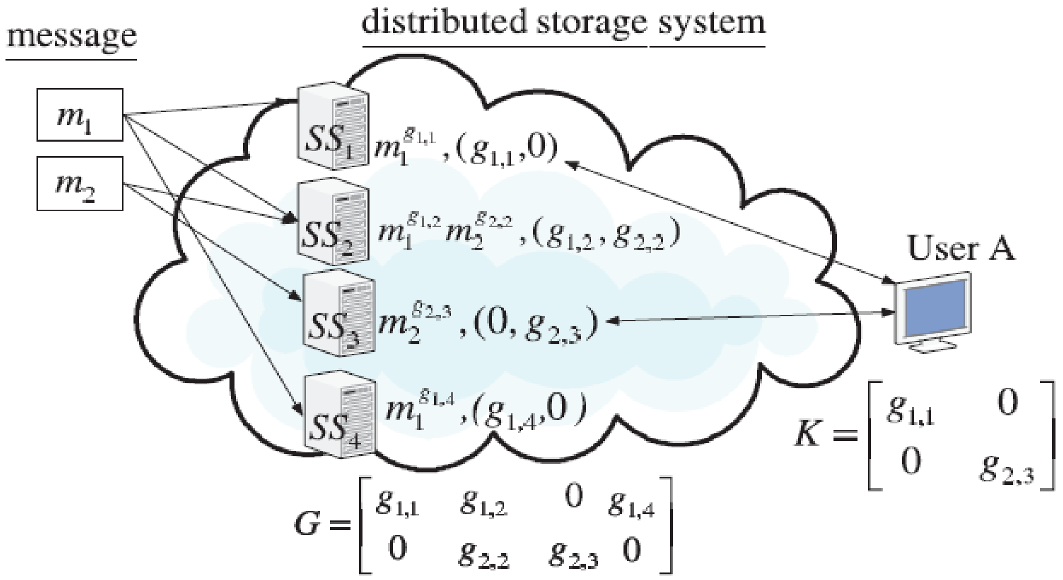
message
distributed storage system
m_1
m_2
SS_1 $m_1^{g_{1,1}}, (g_{1,1}, 0)$
SS_2 $m_1^{g_{1,2}} m_2^{g_{2,2}}, (g_{1,2}, g_{2,2})$
SS_3 $m_2^{g_{2,3}}, (0, g_{2,3})$
SS_4 $m_1^{g_{1,4}}, (g_{1,4}, 0)$
User A
$K = \begin{bmatrix} g_{1,1} & 0 \\ 0 & g_{2,3} \end{bmatrix}$
$G = \begin{bmatrix} g_{1,1} & g_{1,2} & 0 & g_{1,4} \\ 0 & g_{2,2} & g_{2,3} & 0 \end{bmatrix}$

Fig.4 Distributed Storage pattern in cloud

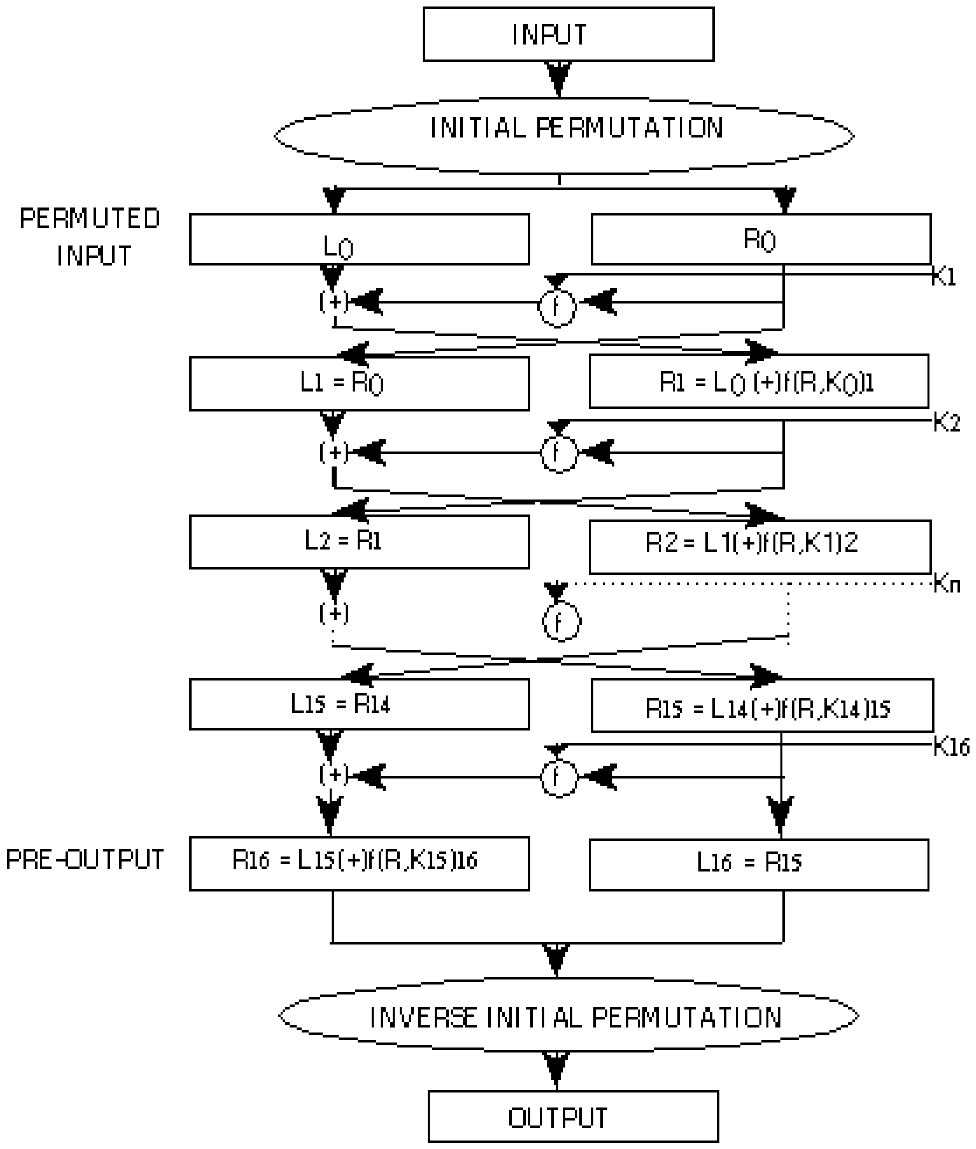

Fig5: DES encryption

REFERENCES

1. A. Haeberlen, A. Mislove, and P. Druschel, "Glacier: Highly durable, decentralized storage despite massive correlated failures," in Proceedings of the 2nd Symposium on Networked Systems Design and Implementation .

2. H.-Y. Lin and W.-G. Tzeng, "A secure decentralized erasure code for distributed network storage," IEEE Transactions on Parallel and Distributed Systems,

3. A. G. Demakis, V. Prabhakaran, and K. Ramchandran, "Decentralized erasure codes for distributed networked storage," IEEE Transactions on Information Theory, vol. 14, pp. 2809–2816, 2006.

4. Q. Tang, "Type-based proxy re-encryption and its construction," in Proceedings of the 9th International Conference on Cryptology in India:Progress in Cryptology - INDOCRYPT, pp. 130–144, Springer, 2008.

5. C. Ungureanu, B. Atkin, A. Aranya, S. Gokhale, S. Rago, G. Calkowski,C. Dubnicki, and A. Bohra, "Hydrafs: a high-throughput file system for the hydrastor content-addressable storage system," in Proceedings of the 8th USENIX Conference on File and Storage Technologies - FAST, p. 17,USENIX, 2010.

6. W. Dong, F. Douglis, K. Li, H. Patterson, S. Reddy, and P. Shilane, "Tradeoffs in scalable data routing for deduplication clusters," in Proceedings of the 9th USENIX Conference on File and Storage Technologies - FAST, p. 2, USENIX, 2011.

7. Z. Wilcox-O'Hearn and B. Warner, "Tahoe: the least-authority filesystem," in Proceedings of the 4th ACM International Workshop on Storage Security and Survivability - StorageSS, pp. 21–26, ACM, 2008.

P. Druschel and A. Rowstron, "PAST: A large-scale, persistent peer-topeer storage utility," in Proceedings of the 8th Workshop on Hot Topics in Operating System - HotOS VIII, pp. 75–80, USENIX, 2