

# Tri-Partite graph: a novel security scheme for cloud data

**P Dileep Kumar Reddy<sup>1\*</sup>, C Shoba Bindu<sup>1</sup>, R Praveen Sam<sup>2</sup>**

<sup>1</sup>CSE JNTUA College of Engineering Ananthapuramu

<sup>2</sup>CSE GPREC Kurnool

\*Corresponding author's e-mail: dileepreddy503@gmail.com

Received 17 April 2017, www.cmnt.lv

## Abstract

Cloud data security is the most concentrated feature of the cloud computing technology. Many cloud computing techniques like cloud data partitioning emerged reflecting new heights of providing data security by defining data priorities. The proposed method presents a novel scheme of maintaining owners prioritized data, while equally ensuring the security for whole portion of the data. The proposed method uses a tripartite graph for securely managing the prioritized data at various levels.

## Keywords:

Encryption,  
Authentication,  
tripartite graph,  
Hash,  
MAC

## 1 Introduction

Today's most elegant data service model is the cloud computing technology that offers tenants with varied shared pool of resources. Wide spread with economical benefits cloud computing succeeded in attracting the major enterprises and totally emerged as widely recognized computing technology. The cloud is intended to provide various hardware and software services where in a user can access these services from anywhere in the world. The applications providing these services need a model of computation, a storage model and a model for communication. The model of computation ensures quality of service for better cloud management. Addressing the important aspect of quality of service is the cloud data security which has always emerged as a vital service by the cloud. To ensure data security the minimum policies expected from the cloud include: a strong encryption mechanism to safeguard the cloud data; prevention of unauthorized access to the cloud using strong authentication schemes ensuring cloud data integrity.

Present research is setting its trend with its innovative, more secured, proactive methods to ensure data security. More than 70% of the cloud providers use either SSL level encryptions or various symmetric key encryption methods for data privacy. Indexes are being geared up to make the search easy over encrypted data in the cloud. Raising the strength of security levels some of the approaches even encrypt these indexes.

A study showed many cloud applications generated the needed keys for data encryption. After data encryption MAC code is generated and appended to the encrypted data to be transmitted along. A thorough study of the prevailing literature showed more than 50% of the cloud service providers followed single mechanism on the whole data. But there may arise a case where the owner at a time is interested with a small portion of his whole data. Even then the cloud mechanism forwards his full encrypted data. This is degrading performance with increased communication cost

and increased application cost.

For efficient data management and accessing the prevailing technique followed by many cloud providers is cloud data partitioning. The owner's whole data is divided into small units and each unit is encrypted and signed so as to raise the data security. Works proved the data partition technique increase the storage efficiency of the cloud as well as error free data retrieval.

The whole of owner data is of not equally important to him. There may be cases where he frequently queries a common portion of data and some parts of the data may be less queried and sometimes not queried at all. This is even apt in cases where a single user data is prioritized as: 1) private- which is the owners sensitive data and in need of strong security mechanisms to protect it; 2) public-which can be accessed by any authorized user and where in case the strength of security mechanisms may be decreased. In the scenarios of clouds, with prioritized data categorizations it is better to have separate security policies running on top of each prioritized level.

The main drawback with cloud data partitioning with prioritization is owners trust. In data partitioning with prioritized data the owner has the provision to query his most prioritized partition. When such data is requested, only that data is securely being forwarded by the cloud. At this moment the owner may feel unsecured with other parts of the data. Though he has proved the data integrity of the queried part he may be more concern with other parts of the data. If there is a mechanism which constantly provides the data integrity of the parts that are not being queried by the owner then it would have raised the owners trust on partitioned data.

This paper presents a novel scheme for maintaining prioritized data of the cloud where a new data structure called the tripartite graph is taken advantage. Using the tripartite graph the strength of security is varied at different levels. The main contributions of the work are:

- A mechanism for easy maintenance of prioritized data.
- Usage of tripartite graph to increase the complexity

of data security.

- Provision to check the integrity of other un-queried data parts.

The organization of the paper is as follows: Section 2 presents the literature reviewed. Section 3 presents the basic preliminaries required to escalate the proposed method. Section 4 presents the proposed approach. Section 5 discusses the performance study and finally Section 6 concludes the paper.

## 2 Literature

The need for organizational growth has made the enterprise to outsource their storage and computing needs. The advent of cloud as a data storage has made the data owner to ponder on how secure the data within the cloud is. To address these security concerns the cloud provider assured security services like:

1. **Owners Data encryption:** So that the storage provider does not learn about owners important data.
2. **Integrity check:** Using which owner's data modification by the provider can be detected.
3. **Secure data sharing:** Authorized users can securely access the enterprise data from the cloud.
4. Efficient data retrieval techniques.

Works discussed in [1] showed the performance of various symmetric key data encryption schemes and showed AES has highest data security capabilities. But the drawback is the data owner faced problems to analyze their own data in the encrypted form. As data owners are naïve to encryption mechanisms they feel that large data encryption makes data loss if not correctly decrypted and so many owners prefer no encryption scheme while storing their data in the cloud. But unencrypted data in the cloud may be a threat. Cloud data security with homomorphism encryption by Adriana and Eran [2] boosted the standards of encryption schemes, where owner can by own analyze his encrypted data. Works presented in [3] discussed various issues of owner on security schemes like:

1. Is all data correctly encrypted.
2. How strong are the encryption algorithms?
3. How strong is the key maintenance mechanism?
4. Even if data is stored encrypted since I am a lame man I can think encrypted data can be decrypted by the cloud and can be changed. How can the cloud raise my trust to address this.

Data integrity is where unauthorized data modification to be identified. Works discussed in [4] showed how an appended MAC code can check data integrity. On data retrieval the owner tries to calculate the MAC and compares with appended MAC; if both are equal the data is not changed. Ensuring data integrity using a third party auditor (TPA) is discussed in [5]. The TPA who is on behalf of owner performed well in verifying the integrity of the dynamic cloud data. Data integrity verifiability using bilinear maps is discussed in [6].

Secure data sharing is where an authorized user can access the cloud data using some kind of user verifiability schemes. A key aggregate scheme for secure data sharing is discussed in [7]. The work discussed an approach of secret key sharing between various data sharing parties where in later stage these keys are used to identify the owner's accountability. Secure cloud data sharing using Tokens is

discussed in [8], wherein a user with appropriate token is only allowed to search the encrypted data of the cloud.

For efficient data retrieval from the cloud techniques like data partitioning are discussed in [9]. An index method is used on the partitioned data for efficient data retrieval. The efficiency of distributed hash indexing on data partitions is discussed in [10]. The other implemented technique for efficient data retrieval is prioritizing the data stored. Works discussed in [11] showed a classification of cloud data levels as public, private there by restricting the private data access with more strong security specifications.

## 3 Background and preliminaries

Cloud technology has envisioned as the present generation cream of hard research. A cloud is a storage area where massive data owners can securely store and can access their data whenever needed. Today cloud computing is more appealing with its high quality applications and services. Though cloud computing is excelling with its computing capabilities, it still faces some new born threats concern to data storage. This section discusses some threats as a part of background study and approaches to counter them. The section also discusses the preliminaries needed to move our proposed work.

### 3.1 PROBLEMS ON HUGE DATA TRANSFER

Huge data transfer between the owner and the cloud is always a bottleneck. Today many of the cloud providers are using link up clouds to transfer owner's data from cloud to owner. These link up clouds are vulnerable to attacks and may raise to huge reliability problems. The Azure cloud of Microsoft underwent with a serious outrage accident where in the owners secured data is lost at the vendors link ups [].

The other problem can be high cost of huge data transfer. Though the owner has stored huge data of his in the cloud sometimes he is interested to only a portion of the whole data. But this is not possible as owners whole data is placed encrypted with his public key and the whole data only can be decrypted by the owner's private key.

### 3.2 PROBLEMS ON HUGE DATA ENCRYPTIONS

Many of the service providers provide no encryption mechanisms as owners feel querying their encrypted data is an increased overhead, the owner may not trust the strength of encryption algorithm. Since the encryption mechanism is done as a part of cloud service; making the owner to have less trust on cloud providers. Since whole data is being encrypted owners are at a fear of losing the entire data if not correctly decrypted and most of the owners prefer to place unencrypted data in the cloud as accessing their direct readable data is easy.

### 3.3 PROBLEMS ON HUGE DATA AUTHENTICATION

Data authentication is a mechanism where the owner is given provision to check the integrity of their stored data in the cloud. A known mechanism for data authentication is appending a MAC to the data. At the verification side the MAC is generated on the data and if appended MAC and generated MAC are

equal the owner's data is authenticated, implicitly showing the data is not altered. Today most of the owners are every time checking the integrity of whole data as their trust on the cloud provider is less. But working the authentication mechanisms even on the unnecessary parts of owner's data is waste of time and raises the cost of the application.

All these problems have risen because of considering the mechanism on the owner's whole data. The proposed mechanism addresses these issues using a naïve structure: a tripartite graph.

### 3.4 CLOUD DATA PARTITIONING

For efficient accessing of the large data, partitioning of bigger data is what followed. Partitioning makes data storage easy at the cloud. When the data is fed at the client module then itself partitioning is done and each partition is encrypted and stored at the cloud. While data is being partitioned many of the cloud providers follow partitioning with various priorities. Data is generally prioritized as sensitive and Un-sensitive.

Sensitive data is where owners most important files reside. The frequency of accessing these files may be more. Un-sensitive data is not of that important to the owner. The frequency of owner accessing these files may be less when compared to sensitive files. Sensitive data may be frequently queried by the owner. Un-sensitive data may be rarely queried or even not queried by the owner. When data is partitioned with such priorities owners trust on his stored data is very much concerned. When he frequently queries his sensitive data, then the data integrity of this portion can be frequently checked there by raising his trust on the cloud. But what about the portions of the data that are not queried or rarely queried. The owners trust on the integrity of these portions may be zero or less. This is the main disadvantage of partitioning with priorities. To address this, the proposed method uses a tripartite mechanism of checking the integrity of un-queried data so as to raise owner trust on the cloud.

### 3.5 PRELIMINARIES

This section discusses some preliminaries required to escalate the proposed work.

**Owner:** The owner is the person who created the data and willing to store the data in the cloud. Much of the research has taken the Owner as a lame man without the knowledge of how actually the privacy mechanisms run. He is at a constant thrive for confidentiality and integrity of his data in the cloud.

**Prioritized data:** The whole data of the owner is not of equal importance to him. There can be priorities of data of his interest. The owner may be more interested with a portion of his data which he frequently queries the cloud. This portion of data of his whole data is prioritized.

**Graph:** A graph is structure which can be used to store data or sometimes used to implement the logic of the procedure. A graph is a collection of nodes.

**Tripartite graph:** A tripartite graph is whose vertices can be divided into 3 independent sets and having an edge between every pair of vertices from independent sets. The tripartite graph divides the whole vertex set under three levels P1, P2, P3. Each level may include any number of

vertices. In our proposed method these vertices are the data files/partitions. Tripartite graphs with 2 vertices in each independent set are denoted by  $K(2,2,2)$  and is as shown in Figure 1:

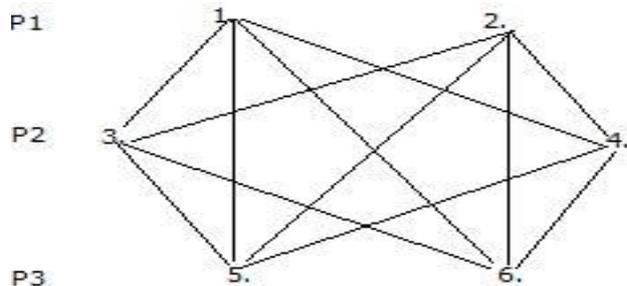


FIGURE 1 Tripartite graph

**2-Partitioning:** If each level of the tripartite graph includes 2 vertices we say the graph is 2-partitioned.

**3-cycle:** We call a 3-cycle as a cycle with its 3 distinct nodes in each of the vertex partitions.

Clearly from the tripartite graph there are 3-cycles joining points of P1, P2, P3. Cycle 1-3-5-1 has its 3 distinct nodes from each vertex partitions. In our proposed work (1,3,5) representing the distinct ends of this 3-cycle is taken as the unique id of this 3-cycle.

Clearly in Figure 1 there are  $2*(2+2)= 8$ , 3-cycles. For a  $K(3,3,3)$  there are  $3*(3+3)= 18$ , 3-cycles. In general in  $K(n,n,n)$  there are  $n*(n+n)$  cycles. An algorithm to generate the possible distinct 3-cycles with unique id is shown.

#### Algorithm 1: Generate 3-Cycles

Input: Tripartite

Output: Distinct 3-Cycles.

Step 1: read the tripartite.

Step 2: for each  $P_i$  ( $i=1,2,3$ ): Start at an arbitrary partition  $F_j$  ( $j=1,2$ ): identify the cycle with its three distinct ends in each  $P_i$ . Give an identity which is the end nodes of the cycle.

Step 3: repeat step 2 for all  $P_i, F_j$ .

Step 4: Store these 3-cycle ids.

### 4 Proposed approach

The proposed method uses a tripartite graph to enhance cloud security at finer granules. A tripartite graph partitions its vertices to be at three levels L1, L2, L3. The proposed method allows the owner to prioritize his data into these three levels, with priority labels P1, P2, P3. Level P1 holds data of high priority like owners sensitive and most personal data; and P3 holds data of less priority. With high priority we mean owner's frequency of querying the cloud for that data is more. Data with low priority P3 is less queried or even not queried. Here we consider a tripartite graph  $K(2,2,2)$ ; shows each leveled data is 2-partitioned with labels F as shown in Figure 2. Owners frequency of accessing F1, F2 of P1 is more when compared to other data. In our proposed approach such a prioritized leveled data is more secured with strongest security specifications. More over our proposed approach provides a way to check the data integrity of un-queried data of the owner.

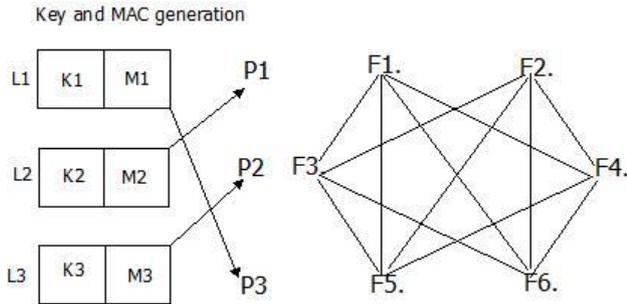


FIGURE 2 Tripartite graph: Varied Specifications

4.1 LEVELS OF SECURITY

The proposed approach discusses levels of security as how, where the strength of security specifications are varying. The proposed work introduced the concept of levels of security as owner always has the high frequency of requesting his high prioritized data at level P1. Which indicates the high prioritized data is frequently communicated in the channel; there by a need for strengthening the security specifications at prioritized level has risen.

The strength of security standards vary as the data priority vary among the 3 levels: P1, P2, P3. Here to increase the complexity of security mechanism, the specifications are used in a way the tripartite edge is connected between the key generation module and data partitioning F. A tripartite edge between L1 and P3 shows the specifications at L1 are used to secure the data at P3 i.e F5,F6. A tripartite edge between L2 and P1 shows the specifications at L2 are used to secure the data at P1. A tripartite edge between L3 and P2 shows the specifications at L3 are used to secure the data at P2.

Since level L1 is connecting less prioritized data of P3, L1 uses less stronger security specifications. Since level L2 is connecting to high prioritized data of P1, L2 uses stronger security specifications. As we know the stronger security specifications are like strong encryption algorithms, strong keys and strong authentication mechanisms.

4.2 SECURITY SPECIFICATIONS

We categorize the security specifications as various mechanisms that provide the data security. The most concentrated security specifications are: Encryptions, Keys and Data Authentication Code.

Stronger specifications: A security specification is strong if:

- The algorithm is very strong with complex modules.
- Difficult to analyze.

Weaker specifications: A security specification is weak if:

- The algorithm need not be that strong, but can solve the purpose.
- Difficult to analyze.

4.2.1 Encryption schemes at various levels

Since level 1 tripartite edge is connecting less prioritized data P3, for data encryption at level 1 less stronger public key cryptosystem, RSA is used. Here F5, F6 are the partitioned data of P3. For better data accessibility, in the proposed approach RSA encryption is done separately on

data of F5 and on data of F6. We denote these encryptions at level 1 as E1(F5), E1(F6).

Since level 2 tripartite edge is connecting high prioritized data P1, for data encryption at level 2 more stronger symmetric key cryptosystem, blowfish is used. Here F1 and F2 are the partitioned data of P1. Blowfish encryption is done separately on F1 and on F2. We denote these encryptions at level 2 as E2(F1), E2(F2). At level 3 for data encryption RSA is used and on the partitions F3, F4 of P2 and are denoted as E1(F3),E1(F4).

Now specifically the encryption specifications are E1-RSA at level 1, 3; E2-Blowfish at level 2.

4.2.2 Data Authentication schemes at various levels

Since level 1 tripartite edge is connecting less prioritized data P3, the defined data authentication specifications at level 1 is MAC: M1; where MAC is of less strong when compared to hash.

Since level 2 tripartite edge is connected to high prioritized data of P1, the defined specifications at level 2 is a hash generated by SHA-512: M2. Since level 3 tripartite edge is connected to P2 of medium priority the authentication specifications defined at level 3 is MD5: M3.

Now specifically the integrity check specifications are M1-MAC at level 1, M2-SHA512 at level 2, M3-MD5 at level 3.

4.2.3 Keys at various levels

To enhance the security of prioritized data the proposed method uses varying key at each prioritized level.

At level 1 as RSA encryption is used, the used key pair is (PR1, PU1). At level 2 since blowfish a symmetric key encryption is used, we use a secret key K1. At level 3 since RSA encryption is used the used key pair is (PR3, PU3). Table 1 shows the security specifications (S) at various levels.

TABLE 1 Security specifications at various levels

Level	S	Encryption	Authentication	Keys
L1	S1	E1-RSA	M1-MAC	PR1,PU1
L 2	S2	E2-Blowfish	M2-HASH-SHA512	K1
L 3	S3	E1-RSA	M3-HASH-MD5	PR3,PU3

4.3 THE APPROACH

This section discusses various storage structures at client and cloud and how data is retrieved from cloud.

Stage 1: Client Storage:

When the owner uploads his data then the client application tripartite the whole data with owners choice of priorities and partitions each accordingly as K(2,2,2). Encrypts each partitioned data as discussed below:

In the proposed approach initially different random numbers are stored at each level. Ensuring less possibility of key cracking these random numbers are assumed to be large. The random number at level 1 is subjected to Fermats Test[] and Miller-Rabin tests[] to generate two primes. These two primes are given input to RSA algorithm to generate PR1, PU1 at level1. Similarly PR3, PU3 at level 3 are generated. Next we have to generate K2 of level 2. Here while K2 is being generated at level 2 the data at P2 and P3 are parallelly encrypted with already generated key pairs of level1 and

level 3 using RSA.

The random number at level 2 is used as a key to run the blowfish algorithm to generate the secret key. The output of the blowfish algorithm is taken as the secret key K2 for encryption at level 2. Once K2 is generated the data at P1 is now encrypted using blowfish algorithm.

Since at the cloud owners encrypted data is stored, owners concern that even this encrypted data can be changed by the cloud is more. To raise owners trust on data integrity, in the proposed approach the integrity check is imposed on top of encrypted data.

The hash, MAC are generated on each encrypted partitions F1, F2 etc and the MAC is appended to each encrypted partitioned data.

**A. Tripartite hash Ring:** Hash is generated on each encrypted partition by giving encrypted data as input to respective hash algorithms at each level as shown in Table 1. This hash is called the tripartite hash and is denoted as  $M(E(F))$  where  $M=M1,M2,M3$  as discussed in section 4.2(B) and  $E=E1,E2$  as discussed in section 4.2(A). All these tripartite hashes of various data partitions at various levels are stored in a client tripartite hash ring as shown in Figure 3.

Now the client stores these tripartite hashes within its tripartite hash ring. The client identifies the possible unique 3-cycles from this tripartite using algorithm 1 and generates a unique 3-cycle id for each and stores with it.

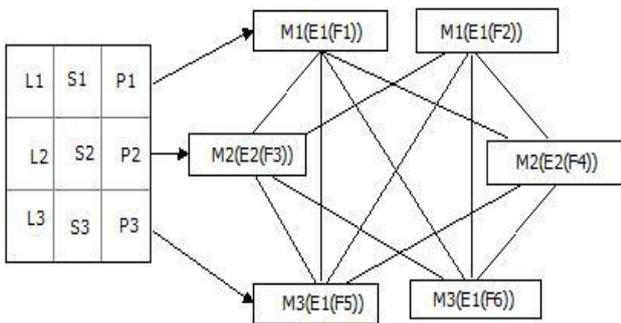


FIGURE 3 Client tripartite hash ring

**Stage 2: Cloud Storage:**

The client module forwards the tripartite to the cloud which includes: encrypted data appended with the MAC and various MAC specifications used at various levels and is shown in Figure 4.

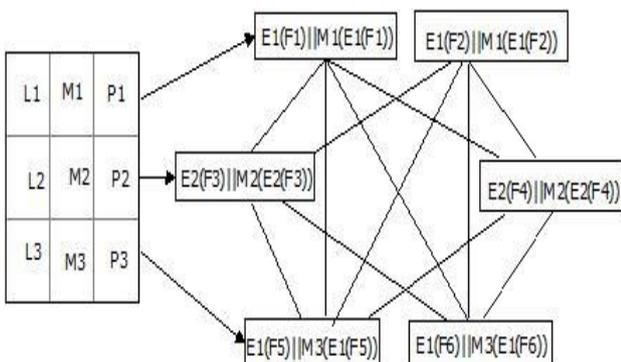


FIGURE 4 Tripartite: forwarded by client to cloud

**A. Cloud verification:** On receiving the tripartite as shown in Figure 4 the cloud performs an initial verification on data modification during the communication from client.

For this the cloud performs hash on  $E_i(F_j)$  and generates  $M_k(E_i(F_j))$ ;  $k=1,2,3; i=1,2; j=1$  to 6. If generated  $M_k(E_i(F_j))$  is same as client appended  $M_k(E_i(F_j))$  then the cloud authenticates data integrity as forwarded by the client.

After verifying the data integrity as forwarded by the client the cloud now constructs its tripartite called the cloud tripartite. It removes each appended hash and stores only each of encrypted data. The cloud tripartite also includes the hash specifications ( $M1,M2,M3$ ) as forwarded by the client and is shown in Figure 5.

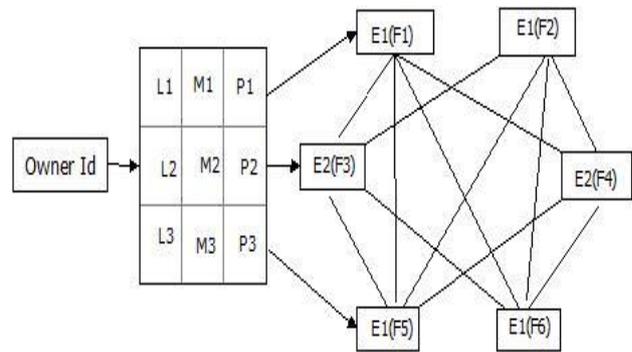


FIGURE 5 Cloud tripartite

**Stage 3: Data Processing**

When the owner queries for the file F1 then the client module sends the following to the cloud:

Client → Cloud:  $L1||P1||E(F1) || Owner\_id$ .

The cloud on receiving this, first process the owner\_id, authenticates the id and indexes to L1, fetches  $E1(F1),M1$  from the storage structure. Using the hash specification defined in M1, the cloud generates hash on  $E1(F1)$  i.e  $M1(E1(F1))$ . The cloud now sends the following to the client:

Cloud → Client:

$1.E1(F1)||M1(E1(F1))||M2(E2(F3))||M3(E1(F5))||(1,3,5)$

Where  $M2(E2(F3)), M3(E1(F5))$  are the hashes generated by the cloud on the encrypted un-queried partitions of the owner. Here  $F1,F3,F5$  are the ends of 3-cycle of the tripartite where these ends  $F1,F3,F5$  are connecting each prioritized level in the tripartite for which  $F1$  is the queried data. So on request of data from priority level the cloud even forwards two hashes from other un-queried less prioritized levels. Here  $(1,3,5)$  is the 3-cycle id forwarded by the cloud.

The cloud may also forward other 3-cycles with  $F1$  as initial vertex; where the first hash refers to the queried data and the rest of the two hashes refer to un-queried data. The other possible 3-cycles starting from  $F1$  are:

2.  $E1(F1)||M1(E1(F1))||M2(E2(F3))||M3(E1(F6))||(1,3,6)$
3.  $E1(F1)||M1(E1(F1))||M2(E2(F4))||M3(E1(F5))||(1,4,5)$
4.  $E1(F1)||M1(E1(F1))||M2(E2(F4))||M3(E1(F6))||(1,4,6)$ .

The selection of 3-cycle from the possible 3-cycles is based on the frequency. If a selected 3-cycle is exceeding its threshold of selection, then another 3-cycle is selected with requested file as the initial point.

**A. Integrity check of the queried partition:**

The owners queried partition is  $F1$ . Now on receiving 1,

the client first process the 3-cycle id (1,3,5), identifies the 3-cycle from its tripartite, the client compares forwarded  $M1(E1(F1))$  with that stored in the client tripartite hash ring. If both hashes are equal  $F1$  is not modified. The client then decrypts  $E1(F1)$  using  $K2$  and finds  $F1$ .

**B. Integrity check of un-queried partitions:**

The client on receiving the other two hashes:

$M2(E2(F3))||M3(E1(F5))$ , compares within its tripartite 3-cycle. Identifies the un-queried data parts from the 3-cycle. If these hashes are equal then integrity of un-queried parts  $F3, F5$  is checked, raising the owners trust on cloud for the un-queried portions. The communication between client and the cloud is shown in Figures 6, 7.

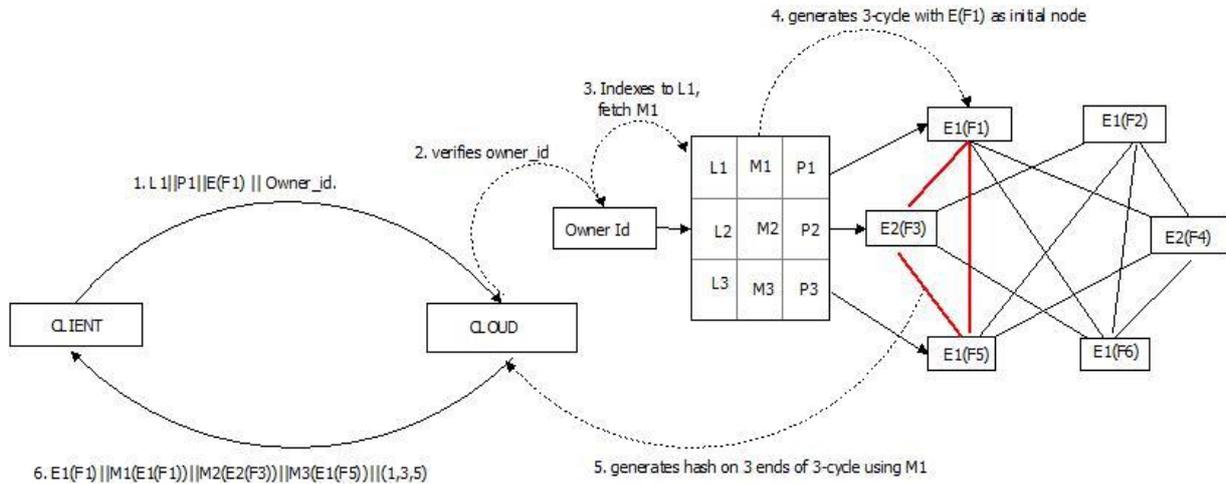


FIGURE 6 Processing by cloud

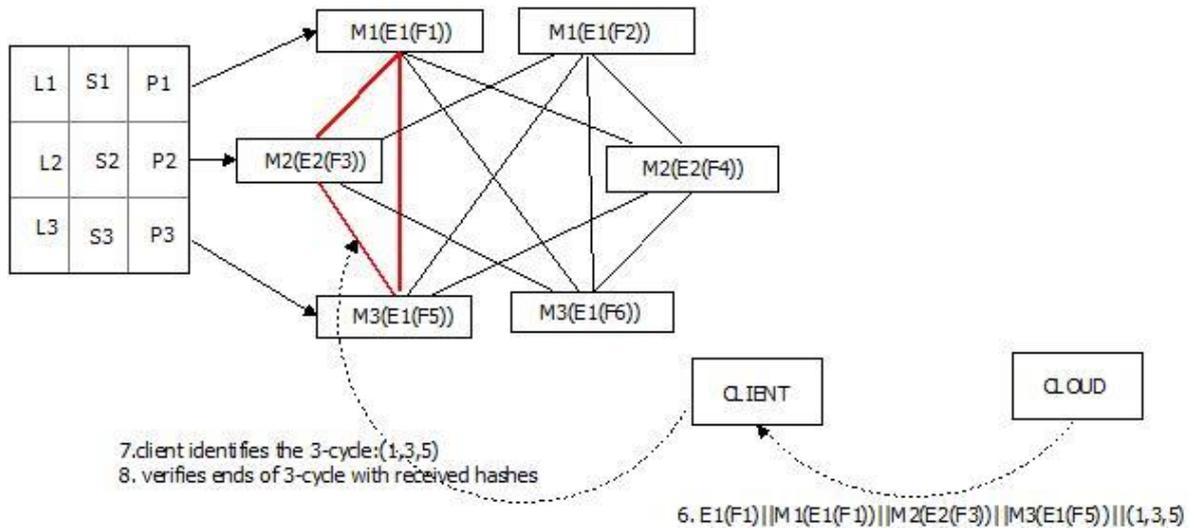


FIGURE 7 Processing by client

**5 Performance comparisons**

The performance study is done under two cases a) Without Partitioning b) With Partitioning.

**A Without Partitioning**

Here the whole data is considered as a single unit. Even data is not prioritized; Owners un-important data is equally treated as that of important data. The encryptions are done on the whole data and the encrypted data is placed in the cloud. If the Owner is interested with just a portion of the data then the mechanism has to decrypt the entire data. Since decryption of data happens at owners client module, the cloud has to pass the entire encrypted data to the client. This may raise communication cost because even the data which is not needed by the owner is passed.

Moreover the time taken to run the encryption and decryption algorithm is high as the whole data is being

considered. A single encryption specification is used for the whole data. Using single encryption specification may increase the threat rate. Even owners un-important data is encrypted at a cost same as important data. This is waste of encryption cost on un-important data.

**B With Partitioning**

The proposed method uses partitioning concept where the whole data is divided into parts under three priorities. There is a provision to query these partitions separately. The frequently queried data of the owner is given top priority. Since the data at this level is important stronger algorithms are used to secure the data at this level. The proposed method then varied the strength of the specifications as the data priorities varied. This type of specification variations at each prioritized level increases the complexity of the method and lowers the threat rate.

The usage of tripartite graph by the proposed method

increased the trust rate of the owner on the cloud. The tripartite mechanism forwarded two un-queried data hashes along with the queried data. With one queried partition the owner can cross check the integrity of two partitions there by the increasing the trust rate by 2% (if there are 100 partitions). Whereas in the usual data partition methods only the queried portion is passed and hence the owner is constantly pondering over the integrity of un-queried partitions thereby reducing his trust on the cloud. Figure 8 shows the trust rate of un-queried partitions.

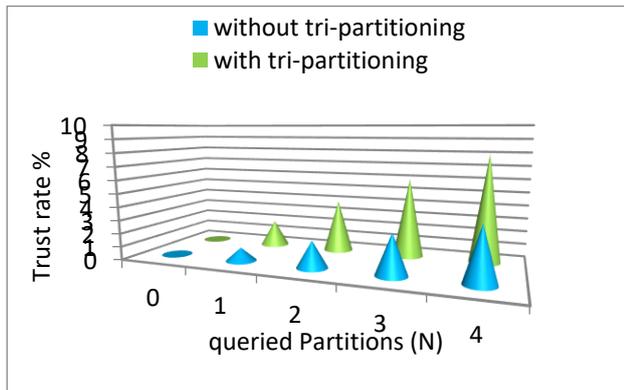


FIGURE 8 Owners trust rate on un-queried partitions.

### 6 Conclusion

Cloud technology has emerged as a boon to enterprise data. A cloud is a storage area where massive data owners can securely store and can access their data whenever needed. Today cloud computing is more appealing with its high quality applications and services. Though cloud computing is excelling with its computing capabilities, it still faces some new born threats concern to data storage and retrieval.

The whole of owner data is of not equally important to him. There may be cases where he frequently queries a common portion of data and some parts of the data may be less queried and sometimes not queried at all. To address this, what followed by many clouds is data partitioning. The owner’s whole data is divided into small units and each unit is encrypted and signed so as to raise the data security. Some clouds partitioned data with various priorities.

The main drawback with cloud data partitioning with prioritization is owners trust. In data partitioning with prioritized data the owner has the provision to query his most prioritized partition. This partition may be frequently queried.

This paper presents a novel scheme for maintaining prioritized data of the cloud where a new data structure called the tripartite graph is taken advantage. Using the tripartite graph the strength of security is varied at different levels there by raising the complexity of security mechanism. Further using the tripartite mechanism the owners trust of un-queried partitions is considerably raised.

### References

- [1] Mohamed E M, Abdelkar H S 2012 Enhanced data security model for cloud computing *IEEE INFOS* 12-6
- [2] Adriana L, Eran T, Vinod V 2012 On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption *ACM-STOC* 1219-34
- [3] Kandukuri B R, RamaKrishna V 2009 Cloud security issues *IEEE-SCC* 517-20
- [4] Sood S K 2012 A combined approach to ensure data security in cloud computing *Network and computer applications* 35(6) 1831-8
- [5] Wang Q, Wang C, Li J, Lou Ren 2009 *Enabling public verifiability and data dynamics for storage security in cloud computing* Springer 5789 355-70
- [6] Wang C, Ren K, Lou W, Li J 2013 Storing shared data in the cloud via security - Mediator *IEEE ICDCS* 124-33
- [7] Chu C K, Chow S S M, Tseng W G, Zhou J 2013 Key aggregate cryptosystem for scalable data sharing in cloud storage *IEEE, parallel and distributed systems* 25(2) 468-77
- [8] Ren K, Wang C, Wang Q 2012 Security challenges for the public cloud *IEEE Internet computing* 69-73
- [9] Khedkar S V, Gawande A D 2014 Data partitioning technique to improve cloud data storage security *IJCSIT* 5(3) 3347-50
- [10] Ye Y, Xiao L, Yen I L, Bastani F 2010 Cloud storage design based on hybrid of replication and data partitioning *IEEE, ICPADS* 415-22
- [11] Kaufman L M 2009 Data security in the world of cloud computing *IEEE security and privacy* 7(4) 61-4

AUTHORS	
	<p><b>P. Dileep Kumar Reddy</b></p> <p><b>Current position, grades:</b> Lecturer in Department of Computer Science &amp; Engineering, JNTUA College of Engineering, JNT University, Anantapur.</p> <p><b>University studies:</b> B.Tech, M.Tech in JNTUA College of Engineering, Anantapur. His areas of specialization are cloud computing, Network Security. He is in teaching since 2010.</p> <p><b>Publications:</b> presented papers at National and International Conferences and published articles in National &amp; International journals.</p>
	<p><b>Dr. C. Shoba Bindu</b></p> <p><b>Current position, grades:</b> Professor and Head of the Department in Computer Science &amp; Engineering, JNTUA College of Engineering, JNT University, Anantapur. She has guided many external and internal projects and has good contributions in many of reputed journals.</p> <p><b>University studies:</b> Ph.D degree in computer science and engineering from JNT University, anantapur.</p> <p><b>Scientific interests:</b> mobile and adhoc networks, network security, data mining and cloud computing.</p> <p><b>Experience:</b> around 16 years of experience in teaching and Research.</p>
	<p><b>Dr. R. Praveen Sam</b></p> <p><b>Current position, grades:</b> Professor in Computer Science &amp; Engineering, G.Pulla Reddy Engineering College, Kurmool.</p> <p><b>University studies:</b> received his Ph.D degree in computer science and engineering from JNT University, anantapur.</p> <p><b>Scientific interests:</b> mobile and adhoc networks, network security, data mining and cloud computing. She has around 13 years of experience in teaching and Research.</p> <p><b>Publications:</b> presented papers at National and International Conferences and published articles in National &amp; International journals. He is life member in CSI, ISTE, IAENG, IE.</p>