Journal of Theoretical and Applied Information Technology
15th July 2017. Vol.95. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A TRIPARTITE PARTITE KEY ASSIGNMENT SCHEME FOR SECURITY OF CLOUD DATACLASSES

Mr. P.DILEEP KUMAR REDDY[1], Dr. R. PRAVEEN SAM[2], Dr. C. SHOBA BINDU[3]

[1]Research Scholar in CSE, JNTUA, Ananthapuramu, A.P, India,
[2]Professor, GPREC, CSE Department, Kurnool, A.P, India
[3]Professor, JNTUACEA, CSE Department, Ananthapuramu, A.P, India
[1]dileepreddy503@gmail.com,[2]praveen_sam75@yahoo.com,[3]Shobabindhu@gmail.com

ABSTRACT

Class based cloud data security is more prominent today. Data of all the classes may not be that frequently accessed by the owner or the users. Data of high prioritized class is frequently accessed and the cloud technology seeks much secured parameters to raise the security of the high prioritized data. More over many of the old cloud technologies needed with user re-authentications when he is simultaneously accessing high class and low class data. This paper presents a new key generation and Class authentication scheme to run the security specifications of the prioritized data. This paper also presents a tri-partite authentication scheme. This scheme has found efficient in reducing the number of user re-authentications.

KEYWORDS: *Cloud Technology, Prioritized Data, Re-Authentications, Tri-Partite*

1. INTRODUCTION

Cloud technology has given a new life to the users of the network by providing a virtual trustable computing environment. Appreciated as a secured platform to store user's data, the cloud is still aiming at improvising the security parameters so as to raise this trust factor. Today's most advanced computing technology has made many organizations to outsource their data storage and computations so as to be relieved from the burden of many economical storage problems. This outsourcing has resulted in two types of cloud infrastructures the private cloud and the public cloud. The data owners maintain the private clouds where they restrict the data to be accessed only by few authenticated users. The public cloud on the other hand is within the control of the cloud service provider and here the owners data is completely out of his control and potentially be used by many unknown users. With public clouds, trust (which is feebly called as security) is the most concerned factor which can be alternatively achieved by defining the best security specifications. Extending business expansions public clouds are preferred. So today much concentration is on enhancing the security of public clouds.

Cutting the edge between the high maintenance cost and data storage, what usually followed by many data owners is partitioning their data to fall into various classes. Literature has revealed the usual data classes are like: private, limited access, public; where private data can only be accessed by the owner, limited accessed data can be accessed by few authorized users and the public data can be accessed by anyone. Data of one class may be frequently queried and other class data may be less queried or sometimes may not be queried at all. With this the cloud is facing the challenge of secure communication of the most frequently queried class. For a cost optimization under such data partitions the cloud technology has varied its security specifications across the classes defined; varied encryption algorithms, varied authentication schemes and varied keys. Out of these varied specifications maintaining and managing the keys needed for data security is the key aspect of cloud technology.

The class based cloud partitioning infrastructure has faced issues in maintaining the keys needed for data encryptions at various classes. For more number of classes more keys are to be generated and managed. Managing these huge lots of keys is a real problem faced by many cloud providers. Literature has showed works where with n number of class's n distinct keys are generated and maintained; which obviously affects the cost factor.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Moreover many of the class based partitioning approaches proposed works where a user authenticated to one class can access only that class with the defined key, but cannot access other class data. But under data sharing users are more inclined to data of other partitions too. Key sharing becomes quite difficult then and also this type increases the number of user re-authentications from class to class.

In this paper we propose a new tripartite class key generation scheme with a novel class authentication mechanism. The owners data is partitioned under three classes: Most Confidential (C1), Confidential (C2) and public data (C3). A tripartite graph connecting the three data partitions is used. The tripartite cycle of the graph is used to authenticate the users and assign keys to various classes. The tripartite approach is designed to strengthen the security of prioritized classes and to reduce user re-authentications. The main contributions of the present work are:

1. Generation of strong random primes to strengthen the encryption schemes

2. Lucas encryptions to strengthen the security of the prioritized classes.

3. A tripartite authentication scheme to reduce the user re-authentications.

The rest of the paper is arranged as follows: Section 2 presents the Literature Survey. Section 3 presents the background and preliminaries. Section 4 presents tripartite class authentication scheme. Section 5 presents the performance analysis. Section 6 presents the conclusion and future extension of the work.

2. LITERATURE SURVEY

With the advent of clouds as storage areas many of data owners are reliant on these clouds for secure data storage. Encryption mechanism is used by the cloud to provide confidentiality to the owner's data. The whole data is encrypted and stored in the cloud. As the whole data is encrypted, while decrypting the approach didn't raise the data owners trust as the owners feel decryption may corrupt their sensitive data which is a part of the whole data. As a promissory step of increasing this trust, what followed by the cloud is data classification: allowing the owner to classify his data to fall into classes like: sensitive and public. Sensitive data can be accessed only by the owner and public data can be accessed by other authenticated users of the cloud. With such data classifications the most concerning factors from the cloud are:

1. How data confidentiality is provided and how users are authenticated to various classes.

2. How keys are generated to access various class data.

3. How these large number of class keys are managed.

Works discussed in [1] presented a set of security protocols called PaaS in the cloud architecture. Paas provided cloud data confidentiality by classifying data into three classes based on the significance of data sensitivity. The class of No-Privacy data is not sensitive and the works didn't use any encryptions on this class data. The other class of data is the privacy with the trusted provider where data is encrypted with the provider's secret key. The third class of data is the Privacy with Non trusted provider where data is encrypted with customers secret key. For each data category the cloud provider allocated a separate storage pool and separate security protocols are used the work showed this kind of data classification enhanced the security of customer sensitive data.

[2] presented a honey pot cloud framework, where the owners data is classified into four classes. The security parameters are then varied according to the sensitivity of the class data. Works discussed in [3] showed a three way classification of the data: Public, Private and limited access based on three cryptographic parameters.

Under such data classification what generally raises is how the keys are generated to access various data classes and how users are authenticated to various classes. A cryptographic key generation scheme for multilevel data security is discussed in [4]. The data classes are defined to form a partially ordered set based on hierarchies. Users with low ordered class hierarchy are not authenticated to access the data with high ordered class hierarchy. Based on these partially ordered levels the class keys are generated to access the data of each

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

class. Works presented in [5] showed a novel attribute based encryption scheme to generate various class keys. The scheme achieved data sharing at finer levels between users authenticated to various privileged classes.

Works discussed in [6] followed a hierarchical access control policy using which data is organized to fall into various security classes. A hierarchical key assignment scheme is used to assign cryptographic keys to various classes. Works discussed in [7] showed a dynamic access control mechanism to various data classes of the cloud. Data re-encryptions and key generations are highlighted in the work using mathematical concepts of bilinear pairings. Works discussed in [8] presented a key assignment scheme for access control in defined hierarchical classes. The approach used a time bound on user access of each class where in a user may be in a class for a period of time. Lucas functions are used to set the time bound on each class and modular arithmetic's with a combination of hash functions is used to generate the keys needed to access each class.

3. BACKGROUND AND PRELIMINARIES

Cloud computing has marked as the most emerging technology inspiring many young researchers to work with it. Cloud is one of the promising technologies to provide security to data stored within it. Every time the cloud is focusing on new data security and accessing approaches in order to raise the owners trust. One such accessing technique to increase the efficiency is storing data in owners predefined classes. Literature has provided with many approaches on how various security mechanisms like encryptions, key generations are addressed on these classes. Though cloud computing is excelling with its computing capabilities, it still faces some new born threats concern to these data classes. 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 the proposed work.

3.1 Problems on data classes

On demand owners request has made the cloud to protect his data under his own defined classes like sensitive data, more sensitive data and non-sensitive data. Segregating sensitive data from the whole data has made the attackers to have an easy eye on this part. This made the cloud to hold an additional burden on countering attacks on sensitive data. The cloud has raised the cost of strong encryption schemes on sensitive data class. Some of the cloud storages varied the encryption schemes with varied classes. With these variations the clouds are facing this overhead of encryption specifications.

3.2 Problems on class key generation

With varied encryption algorithms across varied classes the cloud is in need of huge set of cryptographic keys. If the class data is secured with a symmetric key cryptographic system then n keys are be generated for each of n classes. If a public key cryptosystem is used the n+n keys are to be generated. As the number of data classes increases the number of keys to be generated also increases; which raises the cost factor.

3.3. Problems on class key management

The management of these huge lots of class keys is the major problem faced by many cloud infrastructures. If different users are authenticated to access different class data then the major issue faced is how these different user keys are to be managed to authenticate them to various class data. Users of higher class data may be authenticated to access lower class data. With these concerns many of the cloud infrastructures are lack of efficient key management module.

3.4 Re-authentication- its problems

Re authentication is where a user accessing a secured class is asked for proving himself when he want to access another class of his own data. This is a critical issue in many clouds where the user data is divided into number of classes and the user want access his own other class data being in other prioritized class. At this instance the user is again asked to authenticate. The major problem with re-authentication is handover key management. Key hand over for re-authentication should be very fast. How the keys to authenticate from class to class are managed is a major problem.

In this paper we propose a tri-partite Class authentication scheme so as to reduce the user re-authentications while he is moving from high prioritized class to low prioritized class. To strengthen the data security and for strong key generations we used Lucas sequences. Our paper

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

didn't address on the handover key management issue of re-authentications. This we want to propose as our further extension to tri-partite scheme.

3.5 Preliminaries

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

Owner/user: The owner is the person who created the data and willing to store the data in the cloud. At times he can also be the user of the data. 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.

Class Data: Generally the whole data of the user may not be so confidential. Many of the clouds for better performance are allowing the owner to classify his data into classes like Confidential, Public etc.

Prioritized Class data: Though classified the whole data of a class may not be of equal importance to the owner. There may a portion of the class (file) which he may frequently access. This portion of data of his whole class 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.

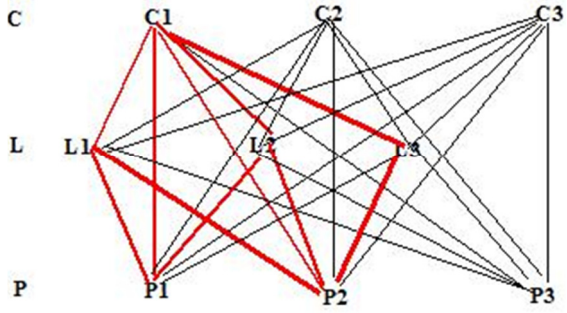

Figure 1: Tripartite graph

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. Each level may include any number of vertices. In our proposed method the three levels are the three data prioritized levels(C, L, P) and vertices are the data files/partitions. Tripartite graphs with 3 vertices in each independent set are denoted by K(3,3,3) and is as shown in Figure 1:

3-Partitioning: If each level of the tripartite graph includes 3 vertices we say the graph is 3-partitioned.

3-cycle: We call a 3-cycle as a cycle with its 3 distinct nodes in each of the three classes.

Clearly from the tripartite graph there are 3-cycles joining classes of C, L, P. Cycle C1-L1-P1-C1 has its 3 distinct nodes from each Class partitions.

In our proposed work (C1,L1,P1) representing the distinct ends of this 3-cycle, is taken as the unique id of this 3-cycle. We call these 3-cycles as tripartite cycles. In our approach we restrict a tri-cycle can only start from high prioritized data of class C.

Clearly in Figure 1 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.

Bi-partite edge: A bipartite edge joins Class L to Class P. Here in our approach we restrict a bi-partite edge can only join L to P data.

Lucas Numbers: The Lucas numbers are defined by

L_n= 2; if n=0;

=1; if n=1;

$=L_{n-1}+ L_{n-2}$; if n>1; for all nE N

A Lucas prime is a Lucas number that is prime.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3-Lucas Partitions: To arrive at 3-Lucas partitions over N we consider 3- partitions of N: [0,N/3) U [N/3,2N/3) U [2N/3, N]. The 3-lucas partitions are defined as:

$L^1_n = L_{n-1} + L_{n-2}$; n E [0,N/3);

$L^2_n = L_{n-1} + L_{n-2}$; n E [N/3,2N/3);

$L^3_n = L_{n-1} + L_{n-2}$; n E [2N/3, N];

Lucas generating function: Let n be an RSA integer where n=pq,a€N. L(a) be the Lucas sequence generated by a, where $L_n(a)=L_{n-1}(a)+L_{n-2}(a)$; the generating function of the above Lucas recurrence is $f(X)= X^2 + aX+1$. We know that the cryptographic strength of second order equations are more when compared to linear. This shows that the Lucas parameters generated from Lucas sequences are cryptically strong.

4. PROPOSED SCHEME

We used Lucas sequences for strong encryptions and for generation of strong keys. A tripartite graph mechanism is used to authenticate users to various data classes. Classify the owners data into three classes: Confidential(C), limited access (L), Public (P). The proposed method allows the owner to prioritize his data into these three levels. Level C holds data of high confidentiality like owners sensitive and most personal data and only the owner is authenticated to this level data; level L holds somewhat owners private data but only few are authenticated to use this level; Level P holds public data of the owner which can be accessed by any cloud user. We assume that these three classes (C, L, P) follow a partial ordering P<L<C. By partial ordering the users of Class C data can access the data at levels L,P also; whereas users of class L,P cannot access data of class C. Users of Class L can access data of class P also. Users of class P can only access data of class P. These three class data is governed by a tripartite graph whose tripartite cycle connects these classes.

Even though the owner has kept his whole confidential data under class C, sometimes he may not be interested with full data. If full data of class C is encrypted then for the requested part of C whole data should be decrypted again. To overcome this headache of full data decryption of each class for a small requested portion of the class, we propose a partite mechanism of partitioning each class data into three partitions C1, C2, C3 of C. The data of C1 is most frequently accessed and will be marked as most confident data and the frequency of accessing data of C2 and C3 is considerably less. Our approach aimed at strengthening the security of C1 to a greater extent when compared to security of data of C2 and C3. One such strong security specification is authentication scheme. Here in this paper we proposed a novel authentication scheme governing user access to each of these classes.

This authentication scheme includes user authentication to access various class data using strong keys. In this authentication scheme we mainly concentrate on the class key generations and authentication scheme for user access. We are not concentrating on class key management, which we want to publish as a future work.

We consider a tripartite graph K(3,3,3); shows each class data is 3-partitioned with labels C(C1,C2,C3), L(L1,L2,L3) and P(P1,P2,P3) as shown in Figure 1. Owner's frequency of accessing label 1 of all classes is more when compared to other data with labels 2 and 3. Here we state label 1 data of all classes is high prioritized. In our proposed approach such a prioritized leveled data is more secured with strongest encryption mechanisms and security specifications. A prioritized authentication mechanism using tripartite cycles of graph is the strength of the paper. In our proposed work we take a central authority (CA) that manages the tri-cycles of the graph.

Each tripartite cycle is generated with three connecting points from each of three classes. We restrict users can access only one partition data either C1 or C2 or C3 or other data at one authentication. Our scheme works with the following specifications:

Initially the CA initiates the user to classify his data into the classes specified by the tripartite mechanism. After the classification the user submits his classified data to the CA. Now the CA runs the security specifications of each class data as follows:

4.1 Generation of Random Primes

If the user want to access class C data then the random primes are generated from the Lucas L^1 sequence: $L^1_n = L_{n-1} + L_{n-2}$; n E [0,N/3);

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

If the user want to access class L data then the primes are generated from the Lucas L^2 sequence $L^2_n = L_{n-1} + L_{n-2}$; n E [N/3,2N/3);

If the user want to access class P data then the primes are generated from the Lucas L^3 sequence $L^3_n = L_{n-1} + L_{n-2}$; n E [2N/3, N].

4.2 Encryption Schemes

We use Elgamal encryption algorithm which is defined by Lucas sequence as follows:

Key pair generation for Class C data: Let p^1,q^1 belongs to L^1_n, n E [0,N/3) ,be Lucas primes, a belongs N. Let Class C public data be (p^1,q^1,a). Select a small integer e belongs to N such that gcd $((p^1)^2-(q^1)^2, d)=1$ and a secret x belongs E [0,N/3); computes d using inverse modulo; computes $y=L_x(a)$;and generates the public key(y,e) and private key(d,x).

Key pair generation for Class L data: Let p^1,q^1 belongs to L^1_n , n E [N/3,2N/3] ,be Lucas primes, a belongs N. Let Class L public data be (p^1,q^1,a). Select a small integer e belongs to N such that gcd $((p^1)^2 - (q^1)^2, d) = 1$ and a secret x belongs E [N/3, 2N/3]; computes d using inverse modulo; computes $y=L_x(a)$; and generates the public key(y, e) and private key(d,x).

Class C Encryption: Since Class C data has three priorities: C1,C2, C3; each partition data has to be encrypted and upload it to the cloud. The encryption has 3 cases:

Case 1: Encryption of C1 data: Since C1 data is more frequently accessed it should be more secured. For this our approach restricts to select a secret k1 in the larger interval between 0<k1< 2n. Uploads the cipherCC1=(CC_1^1,CC_1^2); where $CC_1^1 = L_{k1}(a)$; $CC_1^2 = K + L_e(C1)$ and $K=L_{k1}(y)$.
Decryption of C1 data: When the user wants C1 data then the cloud forwards (CC_1^1,CC_1^2). User computes $K=L_x(CC_1^1)$, $C1= L_d(CC_1^2 - K)$ where (d,x) is his private key.

The same encryption, decryption principles follow for L1 partitions where secrets taken within the longer range [0,2n].

Case 2: Encryption of C2 data: Since C2 data is less frequently accessed than that of C1 data the security of C2 can be lessened. For this our approach restricts to select a secret k2 in a smaller interval between 0<k2<n. Uploads the cipher CC2= (CC_2^1,CC_2^2); where $CC_2^1 = L_{k2}(a)$; $CC_2^2 = K + L_e(C2)$ and $K=L_{k2}(y)$.
Decryption of C2 data: When the user want to C2 data then the cloud forwards (CC_2^1,CC_2^2). User computes $K=L_x(CC_2^1)$, $C2= L_d(CC_2^2 - K)$ where (d,x) is his private key.

The same encryption, decryption principles follow for L2 partitions where secrets taken within the range [0,n].

Case 3: Encryption of C3 data: Since C3 data is very less frequently accessed than that of C2 data the security of C3 can be still lessened. For this our approach restricts to select a secret k3 in a smaller interval between 0<k3<n/2. Uploads the cipher CC3= (CC_3^1,CC_3^2); where $CC_3^1 = L_{k3}(a)$; $CC_3^2 = K + L_e(C3)$ and $K=L_{k3}(y)$.
Decryption of C3 data: When the user want C3 data then the cloud forwards (CC_3^1,CC_3^2). User computes $K=L_x(CC_3^1)$, $C3= L_d(CC_3^2 - K)$ where (d,x) is his private key.

The same encryption, decryption principles follow for L3, partitions where secrets taken within the range [0,n/2].

Case 4: Since data of P is public the data is directly uploaded without being encrypted.

4.3 User authentication to classes

Users authenticated to Class C can also use Class L,P data. Many of the current cloud class authentication mechanisms follow a re-authentications system where in a user if in class C want to access his Class L data then the user has to undertake re-authentication to use Class L data though may be his own data or he is high prioritized user to use other low prioritized data. These re-authentications degrade performance as discussed in section 3.4.In our approach we used a tripartite graph to manage the user class authentications without re-authenticating the user while he is tracing his low class data. A tripartite cycle is used to authenticate the users to various classes according to their priorities.

4.3.1 Generation of signatures

We use Elgamal signatures defined on the Lucas Sequences. For each class data we denote signature as Sig(Class) and define signatures as follows:

Case 1: for class C data the signature is derived as follows: suppose (p,q,a) be the user pubic key. Computes the secrets $Kcp= (a^2 -4)/p$ and $Kcq=(a^2$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

-4)/q. Select an integer x, 1<=x<=(p-1)(q-1), compute $y=L_x(a)$ and make public (n,a,y). Generates the Class C signature as follows: Select a random k such that gcd(k,(p-Kcp)(q-Kcp))=1 and compute $r=L_k(a)$.Compute $s=k^{-1}(C-xr) \bmod (p-Kcp)(q-Kcq)$, where C=C1 or C2 or C3 , k^{-1} is the inverse of k mod (p-Kcp)(q-Kcq).

Now sig(C)=(r,S) is the class C signature where S=Ls(r).

Authentication of Class C signature: The signature is verified if $L^2_C(a) + L^2_r(y) + S^2 - 4 = S\, L_r(y) L_C(a)$.

Case 2: For class L data the signature is derived as follows: suppose (p,q,a) be the user pubic key. Compute the secrets Klp= $(a^2 -4)/p$ and Klq=$(a^2 -4)/q$. Select an integer x, 1<=x<=(p-1)(q-1), compute $y=L_x(a)$ and make public (n,a,y). Generates the Class L signature as follows: select a random k such that gcd(k,(p-Klp)(q-Klq))=1 and compute $r=L_k(a)$. Compute $s=k^{-1}(L-xr) \bmod (p-Klp)(q-Klq)$, where L=L1 or L2 or L3 , k^{-1} is the inverse of k mod (p-Klp)(q-Klq).

Now Sig(L)=(r,S) is the class L signature where S=Ls(r).

Authentication of Class L signature: The signature is verified if $L^2_L(a) + L^2_r(y) + S^2 - 4 = S\, L_r(y) L_L(a)$.

Case 3: For class P data the signature is derived as follows: suppose (p,q,a) be the user pubic key. Compute the secretKPp= $(a^2-4)/p$ and KPq=$(a^2-4)/q$. Select an integer x, 1<=x<=(p-1)(q-1), compute $y=L_x(a)$ and make public (n,a,y). Generate the Class P signature as follows: Sselect a random k suchthat gcd(k,(p-KPp)(q-KPq))=1 and compute $r=L_k(a)$. Compute $s=k^{-1}(P-xr) \bmod (p-KPp)(q-KPq)$, where P=P1 or P2 or P3 , k^{-1} is the inverse of k mod (p-KPp)(q-KPq).

Now sig(P)= (r,S) is the class P signature where S=Ls(r).

Authentication of Class P signature: The signature is verified if $L^2_P(a) + L^2_r(y) + S^2 - 4 = S\, L_r(y) L_L(a)$.

4.3.2The tripartite class authentications

In our approach the tripartite cycles of the tripartite graph are used to authenticate the users to various classes. Our approach uses a CA to manage and maintain these tripartite cycles. As discussed in section 3.1all the possible tripartite cycles are generated and the Ids are saved in the tripartite ring. The CA also saves the Class ids.

The approach

Initially the user identifies his classes of data and partitions them into various partitions. The CA then encrypts the data of classes using various encryption schemes as discussed above. The CA then constructs the signature rings by generating the signatures on each class data. These signature rings are used to authenticate the user from class to class without re-authenticating.

Signature Rings: the CA of the user module generates all the possible Tri cycles connecting the data partitions, generates the cycle ids. The CA then constructs the signature ring which includes the tri-cycles whose vertices are Class|| signature (class). As the data priority varies the structure of signature rings vary. Viewing from class C data the signature rings are tri-cycles. Viewing from Class L data the signature rings are just bipartite edges.

Signature ring of Class C: Since class C data is confidential and a user accessing class C data can also access class L and P data, the signature ring of Class C holds all the tri-cycles connecting the low leveled classes of the partitions.

Signature ring of C1: all the tri-cycles with start vertex at C1, mid vertex at any partition of L and end vertex at any partition of P. Figure 2 shows Signature ring of C1 of C. Similar rings can be constructed for other partitions of C.

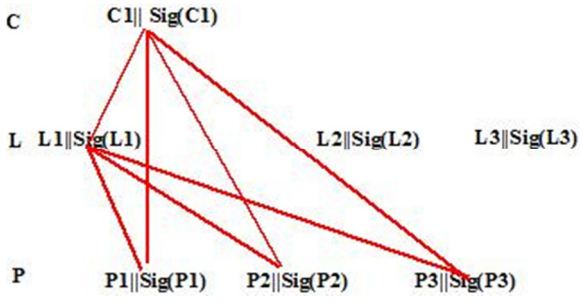

Figure 2: Signature Ring Of C: Collection Of Tri-Cycles

There are a possible of 3+3+3=9 tri-cycles with each of Class C partitions. Hence a possible of nine signature rings from whole C data.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Signature ring of Class L: Since Class L data is with next level of confidentiality, a user in Class L data can only access class P data but not Class C data. So here the signature ring of Class L data includes the bi-partite edges between Class L and Class P data.

Signature ring of L1: all the bipartite edges with start vertex at Class L and end vertex at Class P partitions. Figure 3 shows signature ring of L1,L2,L3 of L.

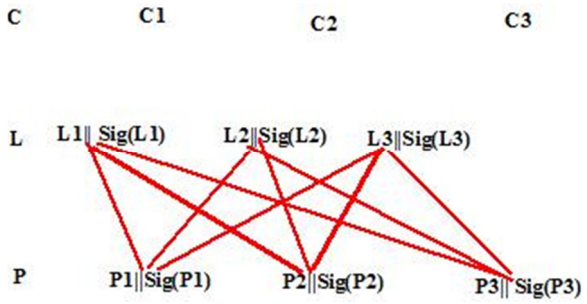

Figure 3: Signature Ring Of L: Collection Of Bi-Partite Edges.

There are a possible of 3+3+3=9 bi-partite edges from each of Class L partitions to class P partitions. Hence a possible of nine signature rings from whole L data.

Signature rings need not be constructed for Class P data as Class P data is public. Here in our approach we have less concentrated on Class C data.

All the possible signature rings are constructed and these signature rings are stored with the CA of the client module. After this the CA uploads the user encrypted data. The CA also forwards the Cloud the tri-partite rings of each class but without the signature as shown in Figure 4.

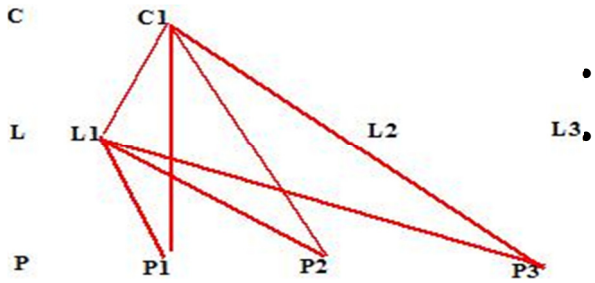

Figure: 4: Signature Rings Forwarded To Cloud

Class authentication procedure:

Now his Class data is C,L,P. Suppose the user want to accessC1 data of Class C. Then in our approach from C1 class the user even can access his less confidential data classes of L without re-authentication. But the user cannot access any other classes of C from C1, though these are his class data to strengthen security of more confidential data C our approach requires re-authentications here.

Here the user puts a request for accessing C1 data. Then the user client module traces the signature ring of C1 from the stored rings at CA, as shown in figure and forwards the cloud user id||Class id(C1)|| signature Ring(C1). The cloud on receiving the request from the user verifies his user id. It then uses the class id to identify which class the user want to access and from Class id(C1) the cloud identifies the user want to access a high prioritized data from which he can use various low prioritized classes. The cloud now traces tri-partite rings of C1 for verification of user signatures. It compares the tri-Cycles of Figure 2 by generating the Lucas signatures at each tripartite end as discussed in section 4.3.1. At C1 it uses user public key, y; to verify

$L^2_C(a) + L^2_r(y) + S^2 - 4 = S\ L_r(y)L_C(a)$ if the signature is matched the cloud authenticates the user to access C1 data. While the user is accessing C1 data the cloud verifies all other low class data signatures by tracing the tri-cycles of the ring of C1. Once the signatures are verified the user is authenticated to use all other low class data if needed without any re-authentications required.

5. PERFORMANCE COMPARISONS

We study the performance comparisons under the following concepts:

- The strength of Lucas encryptions used in our approach.
- The strength of tripartite class authentication scheme in reducing the number of re-authentications.

5.1 The strength of Lucas encryptions

Data security is measured with strong encryption schemes. In other way we can estimate the performance of cloud based architectures analyzing the way they provide data security.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Strong encryption schemes can directly measure the cloud performance. Here in our approach we discuss the strength of Lucas encryptions to support our approach has high performance in terms of data security.

5.1.1 Strength of random numbers

Generally many of the cloud data encryption mechanisms like RSA, ELGAMAL uses pseudo random generators to select the primes that are used for encryptions. The main drawback with Pseudo random generators are they generate primes that are periodic over a small interval. This periodicity is due to the reason that the generators use modular arithmetic. Both bounds of periodicity of modular arithmetic's can be easily estimated. Because of this periodicity the random primes can be easily analyzed.

Our approach used Lucas sequences for generation of random primes. The main characteristic of Lucas sequence is their periodicity is exponential to the initial prime's chosen. Only the lower bound can be estimated. Because of the exponential nature of the sequence it is quite difficult to estimate the upper bound of the periodicity within an interval. Here in our approach we still increases the strength of random primes by varying the interval for each class.

5.1.2 Strength of secret keys generated in the intervals:

Generally many of the cloud security approaches generate public, private key pair as well as secret keys on a common interval like [0,N].Choosing a common interval for all random's and keys makes analyzing them easy. In our approach we have partitioned the interval according to the number of classes. Varying the intervals from which keys are generated may be somewhat tricky. Since there are three classes we have partitioned the whole interval [0,N] into three classes: [0,N/3],[N/3,2N/3],[2N/3,N] from which key pairs of respective classes are generated. Also we have varied the encryption secret keys used by each class within three different intervals. Since C1 of class C data and L1 of Class L are frequently accessed we want to increase the strength of the secrets used for theses classes and so we have varied the secret key (k1) to be taken from the larger interval [0,2n]. The reason behind taking a larger interval is analyzing a larger interval is more secured than analyzing a smaller interval. Since C2 of Class C and L2 of Class L is less frequently accessed, data is not that confidential we varied the secret key (k2) to be taken from a smaller interval [0,n] so as to reduce the computational time. We still reduced the computational time of choosing the secret (k3) for C3 and L3 classes within still smaller interval [0,n/2]. The reason behind taking partitioned intervals for public key pairs and varied intervals for class secret keys is to increase the complexity of analyzing.

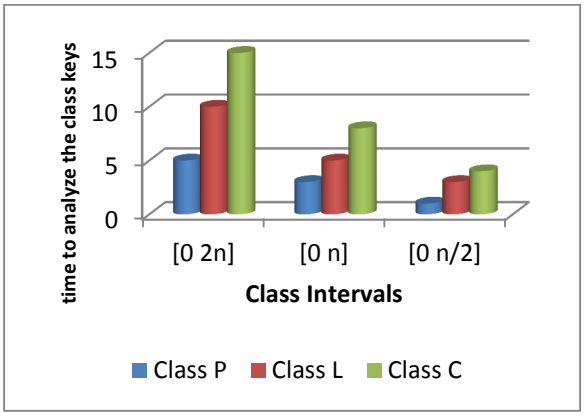

Figure 5: Complexity Of Analyzing Class Keys

With theses varied randoms and varied keys within varied intervals the Lucas encryptions exhibit strong crypto. Figure 5 shows the complexity of class keys within varied intervals.

5.2 Strength of Lucas signatures

Generally many of the cloud architectures use digital signatures generated by some hash functions. The weakness of hash is they are vulnerable to collision attacks. This is because hash functions use linear functions of the type h(x) where a collision happens to be finding x by comparing two similar hashes generated on two different data. Since hashes are linearly convergent theses comparisons are successful in breaking the hash function.

On the contrary in our approach we have used Lucas functions to generate the signature. The Lucas signature is generated using a combination of two functions of the type L(x)+L(y) and hence the complexity of collision attack is very low.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

5.3 Performance comparison of tripartite class authentications

In many of the class based cloud authentications data users are authenticated to access only one class per authentication. If the user want to access some other class being in one class then he has to undergo one more re-authentication. Thus the old class based authentications increases the headache of huge re-authentications. In our approach we used a tripartite graph to reduce these re-authentications. The approach forwarded the tripartite rings of the user where in all the user class signatures are verified when the user is accessing one class data. In the older class based authentications as the number of classes increases the number of re-authentications accordingly increased. Our tri-partite method drastically reduced number of re-authentications irrespective of the number of classes. Figure 6 shows the number of re-authentications in tri-partite class based authentication approach.

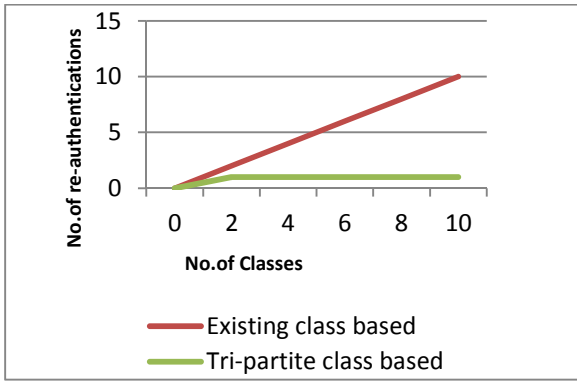

Figure 6: Number Of Re-Authentications

5.4 Strength of our approach

General cloud data encryptions uses secretes, which are randomly picked. Our approach used Lucas sequences which are by nature strong random number generators. Our approach varied the intervals of randoms bases on class priorities and hence can state the Lucas encryptions are more secure. In many of the class based authentications there is no particular approach on how the users are tracing from class to class and hence many of the approaches re-authenticate the users. Our approach used a tripartite graph authentication. The tripartite graph connected these classes and authenticated the users.

6. CONCLUSION

Class based cloud data security is more prominent today. Data of all the classes may not be that frequently accessed by the owner or the users. Data of one class may be frequently queried and other class data may be less frequently queried or sometimes may not be queried at all. This raises the challenge of secure communication of the most frequently queried class. For a cost optimization under such data partitions the cloud technology has varied its security specifications across the classes of priorities defined; varied encryption algorithms, varied authentication schemes and varied keys. Out of these varied specifications maintaining and managing the keys needed for data security is the key aspect of cloud technology. More over many of the old cloud technologies needed with user re-authentications when he is simultaneously accessing high class and low class data. The proposed work uses the strength of Lucas encryptions in generating the class keys and class signatures for user authentications by varying the specifications in various intervals. With theses varied class randoms and varied keys within varied intervals the Lucas encryptions exhibit strong crypto.

A tripartite graph mechanism is used to reduce the number of re-authentications. Our tri-partite method drastically reduced number of re-authentications irrespective of the number of classes.

The present work addressed only on key generation and authentication schemes. Our approach didn't address on managing these keys while authenticating users to various classes. We want to address the key management to various classes as our future extension to the proposed work.

REFERENCES:

[1] WassimItani; AymanKayssi; Ali Chehab "Privacy as a Service: Privacy-Aware Data Storage and Processing in Clou dComputing Architectures", 2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, Pages: 711 &716, DOI: 10.1109/DASC.2009.139

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[2] Md. Rafiqul Islam; Mansura Habiba, "Agent based frame work for providing security to data storage in cloud" 2012 15th International Conference on Computer and Information Technology (ICCIT), Pages: 446 451, DOI: 10.1109/ICCITechn.2012.6509712

[3] Sandeep K. Sood"A combined approach to ensure data security in cloud computing', ELSIVER, Journal of Network and Computer Applications, Volume 35, Issue 6, November 2012, Pages 1831–1838

[4] LeinHarn *, Hung-Yu Lin "A cryptographic key generation scheme for multilevel data security", ELSIVER, Computers & Security, Volume 9, Issue 6, October 1990, Pages 539-546.

[5] DongyangXu; FengyingLuo; Lin Gao; Zhi Tangfine grained document sharing using attribute-based encryption in cloudservers" Third International Conference on Innovative Computing Technology (INTECH 2013), pages: 65 - 70, DOI: 10.1109/INTECH.2013.6653703

[6] Yi-Ruei Chen, CHU Cheng-Kang, Wen-GueyTzeng, Zhou Jianying " CloudHKA: A Cryptographic Approach for Hierarchical Access Control in Cloud Computing", International Conference on Applied Cryptography and Network Security (ACNS), 26 Jun 2013

[7] Ran Yang; Chuang Lin; Yixin JiangEnforcing scalable and dynamic hierarchical access control in cloud computing, 2012 IEEE International Conference on Communications (ICC), Pages: 923 - 927, DOI: 10.1109/ICC.2012.6364473

[8] Jin Li; Xiaofeng Chen; Mingqiang Li; Jingwei Li; Patrick P. C. Lee; Wenjing Lou, secure Deduplicationwith Efficient and Reliable Convergent KeyManagement IEEE Transactions on Parallel and Distributed Systems,Year: 2014, Volume: 25, Issue: 6, Pages: 1615 1625, DOI: 10.1109/TPDS.2013.284

[9] WenGueyTzeny "A time bound cryptographic key assignment service for access control in a hierarchical", IEEE transaction on knowledge Data engineers, Vol 14 No1 , Jan/ Feb 2012.