I J C T A, 9(2) 2016, pp. 717-729

Multi-clouds Computing Security for Distributed Environment Using Differential Evolution Threshold Based Secret Sharing and Data Replication

[1*]N. M. Mallika and [2] B. Srinivasan

ABSTRACT

The development of Cloud Computing (CC) services is catching up speed due to which the rate at which the organizations do the outsourcing of their computational services or carry on their sale of their non-active computational assets. Security is a highlighted concern in cloud and distributed cloud systems. Limited control over the data may incur various security issues and threats which include data leakage, insecure interface, sharing of resources, data corruption, and data availability. During the occurrence of data corruption, the application's QoS requirement cannot have continuous support. In order to provide a clear vision and solving of these problems, this article attempts to identify the chief security issues and data corruption solutions in the field of cloud computing. Data replication is another issue in distributed systems. For the purpose of security, propose a Multilevel Differential Evolution Threshold (MDET) based secret sharing scheme of security for cloud computing, yielding an overview of the status of security currently present in this evolving technology. In MDET enhance the security of secret key in a distributed cloud environment. In the proposed work split secret into multiple shares and store them in different locations using MDET secret sharing scheme. Solve data corruption problem, Multi-Stage Stochastic Integer Programming (MSSIP) is adapted with the intuitive concept of Quality of Service (QoS) to conduct data replication. MDET secret sharing scheme create replicas of secret shares and distribute them among multiple resource providers to ensure availability. An experimentation result shows that the proposed MDET based secret sharing scheme, in addition to identifying the related vulnerabilities and threats along with feasible solutions when compared to existing state of art methods.

Keywords: Cloud Computing (CC), Security, data-intensive application, Quality of Service (QoS), data replication, Multilevel Differential Evolution Threshold (MDET), Multi-Stage Stochastic Integer Programming (MSSIP).

1. INTRODUCTION

The significance of Cloud Computing is seeing a remarkable increase and it is achieving a rapidly grabbing attention in the communities of science and industries. A research conducted by Gartner [1] regarded Cloud Computing to be the first among the top 10 of the most strategic technologies and having better chances in years to come to be employed by organizations and companies. Cloud Computing seems to be a computational model and also a distribution architecture and its chief goal is providing safe, quick, convenient mode of data storage and net computing service, having all the computing resources treated as services and thereafter conveyed over the Internet [2]. The cloud improves co-operation, agility, scalability, availability, the capability to adjust to variations in accordance with demand, speed up the development work, and renders the power for reduction in cost by means of optimized and effective computing [3].

[1] Assistant Professor, Department of Computer Science, Sri Vasavi College(SFW), Erode-638316, Erode District, Tamil Nadu

[2] Associate Professor, PG & Research Department of Computer Science, Gobi Arts & Science College, Gobichettipalayam-638476 Erode District, Tamil Nadu.

**Corresponding Author: E-mail: mallika.scholar@gmail.com*

Distributed cloud [4] makes use of resources provided by users in a Peer to Peer (P2P) manner. In the distributed cloud resources are virtualized. The existing cloud uses huge data centers where resources are provided using virtualization techniques. Some of the voluntary computing systems such as Berkeley Open Infrastructure for Network Computing (BOINC) [5] and Planetlab [6] use resources provided by users, but these are managed by a centralized entity and are entirely different from either a central cloud or else the distributed cloud. Unlike users of the existing cloud, who don't have control over the resources, in the distributed cloud users can choose resources of their choice. Moreover, in the distributed cloud, resources are provided by users. Since users of the distributed cloud have more control over resource selection, they can perform strong encryption to secure their data.

Eventhough there are several advantages to following CC, there also exists few considerable hurdles for adoption. One among the most important roadblock for adoption is security, then followed by challenges related to compliance, privacy and legal concerns. Security is one of the key necessity for the consolidation of cloud computing in the form of a reliable and practical multipurpose solution. This perspective is shared by several unique groups, consisting of academia researchers [7], business decision makers and government organizations [8]. The multiple similarities present in these viewpoints are indicative of a critical concern over essential security and the legal barriers for cloud computing, that includes availability of service, data confidentiality, provider lock-in and reputation fate sharing [9]. These issues have their source not only from the existing challenges, that are inherited directly from the technologies adopted, but also have their relation with new issues that spring from the collaboration of important cloud computing characteristics such as scalability, resource sharing and virtualization (e.g., data leakage and hypervisor vulnerabilities). The only issue will be managing these keys. Nonetheless, the application's QoS requirement is not considered in the data replication. During the occurrence of data corruption, the QoS requirement pertaining to the application cannot be maintained in a continuous manner.

It is hard to enquire all the nodes having the same performance and capability in their CPUs, memory, and disks[10] due to the presence of a huge number of nodes in the system of cloud computing. For instance, the Amazon EC2 is a practical heterogeneous cloud platform that offers different types of infrastructure resource aiming to satisfy various user requirements in the computing and storage resources. The system of cloud computing has heterogeneous features in nodes. Because of the heterogeneity of the node, the data belonging to a high- QoS application might get duplicated in a node with low-performance (the node having slower communication and latencies with respect to disk access). At a later time, in case data corruption happens in the node that is running the high-QoS application, the data belonging to the application will then be retrieved back from the node with low-performance. As the low-performance node is slower in communication and has disk access latencies, the QoS requirement with respect to the high-QoS application may then be broken.

To solve security and data corruption problems, Secret sharing schemes can mitigate the risks of managing these keys. The main ides behind the threshold secret sharing schemes, information is split into multiple shares and these shares are distributed among multiple users. The only way to retrieve the information back is to have all the shares or a qualified number of shares available. To solve data corruption or data leakage problem Multi-Stage Stochastic Integer Programming (MSSIP) is introduced to solve QoS-aware data replication issue for applications that are data-intensive in the cloud computing systems. The data replication issue is concerned with how efficiently the QoS requirements related to applications can be considered in the data replication. The important objective of the data replication issue is the minimization of the cost incurred in data replication and also the number of QoS violated data replicas. By minimizing the cost of data replication, the data replication can be quickly finished.

The remaining part of the paper is organized as the following sections: Section 2 introduces the existing methods for security in distributed environment and preliminaries of existing methods. Section 3 presents data replication problem with security is defined. Section 3 presents secret sharing scheme and data replication

methods are explained in detail. Section 4 provides the evaluation of the performance of the algorithms proposed. At last, Section 5 presents the conclusion of the paper.

2. RELATED WORK

In [11] is multi-cloud model defined as the combination of different clouds in which user data will be distributed and then executed in those clouds at the same time. It is seen that multi– clouds enhance the performance that is rendered by single cloud environment by means of division of security, trust and robustness between diverse clouds. A survey has been made of different methodologies that are available for the security of multi cloud such as usage of cryptography, secret sharing algorithm, DepSky system, Redundant Array of Cloud Storage (RACS) and HAIL protocol. Limitations of existing solutions have been indicated and secure cloud database has been suggested for future work though the solutions had not been given in details.

In [12] introduced the multi-cloud computing framework making use of proxy VM instance for resources sharing and dynamically collaborating among the cloud based services. This framework provides security management, mutual trust a policy problems without needing any pre-collaboration agreement that is needed in cloud mashups. At whatever time, the cloud user wants to make use of any services, he will be sending request to the cloud where the CSP is equipped with pre-installed proxy VM that will interact with multicloud services and provide the user with results. It assists in collaboration between the different cloud user. They have rendered various proxy architectures in addition with cloud hosted proxy, proxy as a service and on premise proxy out of which the first two architecture are dependent on cloud service provider and proxy service provider that cannot solve the problem of malicious system administrator. Peer -to Peer (P2P) proxy architecture will have more security in which the clients control the proxies.

In [13] have proposed four different architectures for the paradigm of multi-cloud computing for enhancing security and privacy level to users and providers. The first approach deals with the application replication that assists in verifying the data integrity after the completion of the execution. The second approach aids in protecting the data and logic by isolating them. The third approach preserves the data and application confidentiality by means of breaking the application logic into parts and then having it executed over multiple clouds. The same kind of approach is provided in the final architecture in which the data is broken down into parts and thereafter executed over different clouds that enables them to safeguard from cloud service provider who are malicious. Every architecture is endowed with its own benefits and de-merits but the combination of this architecture will yield a better approach with security for multi-cloud systems.

Secret sharing schemes have recently been considered to apply for cloud computing in which many users distribute multiple data to servers. However, when Shamir's (k, n) secret sharing is applied to cloud systems, the amount of share increases more than n times the amount of the secret. Therefore, propose a new secret sharing scheme [14] that can reduce the amount of share different from Ramp type secret sharing, suitable for cloud systems, and prove that it is computationally secure.

In order to efficiently assist the cloud storage, a Distributed File System (DFS) has to be many-sided with awesome features in various important viewpoints and with no notable disadvantages. The significant design objectives of a DFS in Clouds are inclusive of security, reliability and scalability. Conventionally, cryptography, data duplication and machines with great power are general approaches for supporting DFS. Nonetheless, the success of DFS such as this will rely on meticulous key management, huge storage and expensive infrastructure, correspondingly. In [15] Blakley's secret sharing is introduced and it is applied to a DFS for both the causes of security and robustness without having to sacrifice over the scalability in the performance considerably. A DFS is implemented with Graphics Processing Unit (GPU) in the form of an acceleration choice for tackling further with scalability issue. Experimental outcomes have exhibited the efficiency of the new DFS.

In [16] N-Cloud scheme is proposed that enhances the availability, performance, and the confidentiality in cloud storage. N-Cloud yields the availability through the division/splitting of a file into several chunks, and then replicating these chunks in a non-overlapping way into several cloud storages, on the basis of considering both security and dependability. In this technique, the chunks present in a file can be encrypted simultaneously when the chunks that are encrypted can be uploaded to geographically isolated cloud storages. Same advantage can be attained for the process of downloading and decrypting. It greatly accelerates the processing along with transfer. N-Cloud offers confidentiality by means of the division/splitting of a file into several chunks, encrypting all the chunks, and then distributing these chunks to different clouds. N-Cloud guarantees that not one of the cloud contains the entire file; the scheme will not have any compromise until an adversary barges into all the cloud storage.

Cloud Seal [17] is a technique for the sharing and the distribution of data securely by means of cloud-based data storage and content delivery services (e.g., Amazon S3 and CloudFront). CloudSeal assures the confidentiality regarding the content that is stored in public cloud storage services, through the encryption of it just prior to sharing over the cloud. Aiming to have access control policies that are flexible, CloudSeal further employs k-out-of-n secret sharing in addition with broadcast revocation techniques for renewing shared secrets, for instance, when a user joins in or leaves out of a content sharing group. The most important fact is that CloudSeal has the leverage over the proxy re-encryption algorithm for transferring part of the stored cipher content in the cloud that can be decrypted by means of a validated user having the updated secret keys. Achieve this property without any modification to most of the content that is encrypted. This feature is significant for the effectiveness of content distribution. All of these methods data corruption happens in the node that is running the high-QoS application, and the data corresponding to the application will then be retrieved back from the low-performance node. Since the low-performance node has slow communication and disk access latencies, the QoS requirement of the high-QoS application may be violated.

3. PROPOSED DIFFERENTIAL EVOLUTION THRESHOLD BASED SECRET SHARING AND DATA REPLICATION METHODOLOGY

In the distributed cloud model [4,18], users do computation and store data in resources provided by other users as shown in Fig. 1. Since users use resources provided by other users, there are obvious security concerns. Dependable Storage in the Inter-cloud [17], provides reasons why single cloud is not secure, namely, need to have more than one cloud to make the system more secure. The Distributed cloud model by itself solves the problem of single domain cloud as it has more than one user who will be acting as a cloud provider. In distributed cloud, protecting data stored on other users' resources poses security issues, which is handled using standard encryption techniques, which in turn leads to key management issues and insider attacks. A resource provider can get access to the data stored on his resources with ease and can use brute force attacks on it. Security of data storage currently depends on how strong encryption keys a user has used or how effective the key management schemes used are. So instead of having a single key that gives access to the encrypted files, propose using multiple shares. In this proposed secret sharing mechanism in the distributed cloud use secure mechanisms to enhance the security of secret key shares. In this model fora server having less space for replication, in case there are several data object replicas that has to be saved in this server, the replicas of few data objects cannot be saved in a successful manner. Here, the data object replicas that are unsuccessful will be placed in other servers with no QoS guarantee. This issue is not dealt with in the above earlier work.

To solve this problem, Secret sharing schemes can mitigate the risks of managing these keys. The main idea behind the Multilevel Differential Evolution Threshold (MDET) schemes, information is split into multiple shares and these shares are distributed among multiple users. The only way to retrieve the information back is to have all the shares or a qualified number of shares available. To solve data corruption or data leakage problem Multi-Stage Stochastic Integer Programming (MSSIP) is introduced to solve QoS-aware

data replication issue for the case of applications that are data-intensive in cloud computing systems. The data replication problem concerns how to efficiently consider the QoS requirements of applications in the data replication. The main goal of the data replication problem is to minimize the data replication cost and the number of QoS violated data replicas. By minimizing the data replication cost, the data replication can be completed quickly.

The Distributed cloud is formed using the resource provided by users and can provide resource for free to everyone who is part of the system. The Distributed cloud makes use of virtualization to allocate resources to multiple users and shares available resources efficiently and it also avoids single point of failure. In this work propose a MDET scheme to protect the secret key shown in Fig. 2 in the distributed cloud environment. The proposed MDET scheme encrypts the file using the secret key before distributing the key shares among participant resource providers whom assume to be honest. At the first level the user splits the key and distributes the shares among resource providers. Instead of attaching key shares as metadata to the pieces of data [9] split the key shares at each resource provider again into multiple shares in the second level. The second level of our mechanism improves the security since to get the original secret the attacker has to have all the shares from the two levels. Generate the threshold value in the second level dynamically which enhances the security as the attacker cannot know about the threshold value beforehand. In addition to that at each resource provider we create dummy keys which increase the probability of knowing if a resource provider is compromised by any attacker.

Secret key generation using the Differential Evolution (DE)

Generation of the key values for user in the Distributed cloud becomes very difficult, so in this work refer a Differential Evolution method is used to generate secret key values for each user in the distributed cloud. Let us consider multiple users $CU = (cu_1, \ldots cu_n)$ and shares available resources providers $RP = (rp_1, \ldots rp_n)$, the key generated from Differential Evolution [19] as $SK = (sk_1, \ldots sk_n)$, the secret key values is splitted into key shares $sk_1 = (sh_{1_1}, \ldots sh_{1_n})$. The data stored of the each user file is denoted as $D = (d_1, \ldots .d_n)$. To generate key values the randomly generated prime positive integers is considered as input to each cloud user *CU*. DE begins with a prime positive integers (population) of NP candidate solutions that may be denoted as $x_{i,G}$, $i = 1, \ldots, NP$, where i index represents the population and G stands for the generation to which the prime positive integers belongs. The operation of DE is dependent on the manipulation and effectiveness of three important operators; mutation, reproduction and selection which are briefly explained in this section.

Mutation: The mutation operation of DE [20] applies the prime positive integer's vector differentials between the available prime positive integers population members for the determination of both the degree and the direction of disturbance that is applied over the individual prime positive integer's subject for the operation of mutation. The mutation procedure at every generation starts by the random selection of three prime positive integers in the population.

$$V_{i,g} = X_{r_1,g} + F * (X_{r_2,g} - X_{r_3,g}) \quad (1)$$

$r_1, r_2, r_3 \in \{1, \ldots, NP\}$ are randomly selected prime positive integers and satisfy: $r_1 \neq r_2 \neq r_3 \neq i$, $F \in [0, 1]$.

Crossover: When the mutation phase is finished, the activation of the rossover process is initiated. The perturbed prime positive integers individual, $V_{i,G+1} = (v_{1,i,G+1}, \ldots, v_{n,i,G+1})$, and the current perturbed prime positive integers population member, $X_{i,G} = (x_{1,i,G}, \ldots, x_{n,i,G})$, are subjected to the crossover operation, which at last does the generation of the perturbed prime positive integers population of candidates, or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{n,i,G+1})$, as below,

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & if\ rand_j \leq C_r \forall j = k \\ x_{j,i,G} & otherwise \end{cases} \quad (2)$$

Where, $j = 1 \ldots n, k \in \{1, \ldots,$ Where, $j = 1 \ldots n, k \in n\}$ refers to the index of a random parameter, selected once for every *i*, and the crossover rate, $C_r \in [0, 1]$, the other control parameter of DE, is determined by the user.

Selection: The population for the next subsequent generation is chosen from the individual in the current perturbed prime positive integer's population member population and its related trial vector in accordance with the rule that follows:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & if\ f(U_{i,G+1}) \leq f(X_{i,G}) \\ X_{i,G} & otherwise \end{cases} \quad (3)$$

In MDET scheme ensure the security of the secret key in a distributed cloud environment. To do that we use the multilevel threshold secret sharing scheme (t, n) in which the first level uses the threshold t < n and the second level uses the threshold t = m. Here the fitness is verified to uses the threshold t < n, then the $f(X_{i,G})$.

Thus, each prime positive integer (individual) belonging to the temporary (trial) population is then compared with its equivalent present in the current prime positive integer's population. The one having the lesser objective function value will then survive from the next selection of tournament towards the next subsequent generation's population. Consequently, all of the individuals belonging to the next generation are nearly as good as or better than their equivalents in the present generation. In the case of DE, the comparison of the trial vector is not done with all of the prime positive integers (individuals) in the present generation, but only with one prime positive integer (individual), its equivalent, in the current generation.

Algorithm 1: Key generation using DE algorithm

Input: Number of cloud users $CU = (cu_1, \ldots cu_n)$, Random prime positive integer $PN = (p_1, \ldots p_n)$

Output : Secret Key $SK = (sk_1, \ldots sk_n)$

Step 1: The first step indicates the random initiation of the prime positive integer *PN* population. Generate randomly a positive integer *PN* population of (for ex) NP vectors, each having n dimensions:

$$x_{i,j} = x_{min.j} + rand\ (0, 1)\ (x_{max,j} - x_{min,j}) \quad (4)$$

where $x_{min,j}$ and x_{max} are the respective lower and upper bounds prime number for j^{th} component (cloud user), rand (0,1) refers to a uniform random number in the range between 0 and 1.

Step 2: Compute the objective function value $f(X_i)$ for all X_i.

Step 3: Select three prime positive integers from the population and generate the perturbed individual V_i employing equation (1).

Step 4: Recombine each target vector x_i with perturbed prime positive integer individual that is generated in step 3 for generating a trial vector U_i making use of equation (2)

Step 5: Check if every variable of the trial vector lies within the range. If yes, then go to step 6 else bring it within the range making use of $u_{i,j} = 2 * x_{min,j} - u_{i,j}$,if $u_{i,j} < x_{min,j}$ and $u_{i,j} = 2 * x_{max,j} - u_{i,j}$, if $u_{i,j} > x_{max,j}$, and go to step 6.

Step 6: Compute the objective function value for vector U_i.

Step 7: Choose better secret key value out of the two (function value at target and trial point) applying equation (3) for the next generation.

Step 8: Check if the convergence criterion is satisfied, if yes then stop; else go to step 3.

Differential Evolution method is used to generate secret key values for each user in the distributed cloud. The user splits the secret key into n number of shares S_i, where i ∈ (1, n) and distribute them among all resource providers. In this level the threshold value which indicates that to reconstruct the secret key [21] at least t number of shares would be required. Each share of secret S_i is replicated into k numbers so that if one resource provider goes offline or compromised then that share can be accessed from other resource provider. At each resource provider further split the share S_i into m number of shares S_{ij}. In this stage store the data file of each user in distributed cloud. Here if any data loss occurs it have been selected from another source using the data replication schema. Multi-Stage Stochastic Integer Programming (MSSIP) [22] is introduced to solve QoS-aware data replication problem for data-intensive applications in cloud computing systems. At this level, i.e., second level the threshold value m is generated dynamically. To determine the number of shares m each resource provider Rp_i, i ∈ (1, n) selects a (P_i,N_i) pair from the share pool SP. The share pool is created beforehand. The user saves the pair (i, P_i) for each provider RP_i and the provider saves N_i. Intend to have multiple share pools and place one or two of them in each cluster of the distributed cloud. The CRT solution generates a number m which decides the number of shares to split and reconstruct in the second level. At each resource provider at least j number of dummy shares SD_{ij} ,where i ∈ (1, n) and j ≥ m, are generated. Whenever a resource provider RP is compromised, the user revokes the access of that particular resource provider. Intend to have a greater number of dummy shares than secret shares, i.e., j ≥ m, so that if any outside attacker tries to get the share, the probability that he ends up with the dummy share instead of a real share is greater than or equal to 0.5. This helps the user to take action (e.g., revoke the access of that resource provider) accordingly when some attacker selects a dummy key. To reconstruct the key sub shares, each resource provider need to have the P_i from the user to generate the threshold value m. The user reconstructs the secret S from t numbers of S_i shares.

Algorithm 2: Multilevel Differential Evolution Threshold (MDET) schemes

Input: Number of cloud users $CU = (cu_1,... cu_n)$, Random prime positive integer $PN = (p_1,... p_n)$, shares available resources providers $RP = (rp_1,... rp_n)$, the key generated from Differential Evolution as $SK = (sk_1,... sk_n)$, the secret key values is splitted into key shares $sk_1 = (sh_{1_1},...sh_{1_n})$. The data stored of the each user file is denoted as $D = (d_1, ...d_n)$.

Output : Key sharing

1. for $CU = (cu_1, ... cu_n)$ do
2. Generate prime positive integer *PN* from the *DE* and generate key values *SK*
3. User CU encrypts the file D with secret key SK
4. Split the secret key SK into n number of shares, Sh_1, Sh_2, Sh_3, Sh_4, .., Sh_n.
5. To reconstruct SK at least t number of shares is required.
6. for each share i of S, where i ∈ 1, n do
7. Replicate the share Sh_i into k ≥ 1 number of replicas.
8. Distribute the replicas among n number of resource providers.
9. If any data loss occurs call data replication MSSIP
10. end for
11. for each resource provider, RP_i, i ∈ 1, n do
12. Select a pair (P_i,N_i) by the following steps
    - for i = 1 atleast n do

- Generate a random series of pairwise relatively prime positive integers, $P_i = p_{i1}, p_{i2}, ., p_{im}$
- Generate a random series of m arbitrary integers $Ni = n_{i1}, n_{i2},..., n_{im}$.
- Place these two series Pi along with Ni, represented as (P_i,N_i)
- end for

13. Cloud User CU saves (i, P_i) and RP_i saves N_i
14. Get a unique solution $m = x_i$ from (P_i,N_i)
15. Split the share of secret Sk_i into m number of shares, $S_{i1}, S_{i2}, S_{i3},S_{i4}, .., S_{im}$
16. Generate j number of dummy shares $S^D_{ij,}$ where $j \geq m$
17. Reconstruct the share of secret Sh_i from m number of shares,
18. end for
19. for each resource provider , Rp_i, $i \in 1, t$ do
20. Collect the share Sh_i from each resource provider
21. end for
22. Reconstruct the secret key SK from Sh_i where i = 1, ..., t.
23. end for

Next we present replication algorithm for solving the data replication issue in the cloud computing system. Before detailing over the proposed algorithm, first few definitions are given for explaining the problem of data replication. Given a distributed cloud computing system having a set of storage nodes SN, they can also have applications running along with storing data. The storage node functionality is just as the same as the storage node in HDFS [23]. For a given storage node $sn \in SN$, if its application that is running writes over a data block db to the disk of r, then a request for replication will be issued from r to have a number copies of b replicated to the disks belonging to other nodes. |SN| is generally large, in the cloud computing system. It is very much possible that there may be several simultaneous replication requests that are issued from various nodes at a particular time period. Because of limitation in space, every node cannot have too much of data replicas stored from other nodes. For the case of a data block db, in case it gets replicated from node s to node q, one of the data replica dr corresponding to b will then be stored in q. A suitable access time T is then indicated for dr. Additionally, dr has a cost of replication RC along with an access time AC. If the actual copy of b cannot be read because of data corruption, r tries to get the data replica dbr retrieved from q. In case AC becomes greater than T, thern dbr is a QoS-violated data replica. Table 1 tabulates the notations employed in data replication algorithm.

In case an application wishes to write a data block, then the node that is executing the application would make a request of replication for that data block. The information regarding the application's QoS requirement (the access time desired for the data block) also is attached on the replication request for the purpose of generating a QoS-aware replication request. Several number of QoS-aware replication requests might be issued simultaneously from multiple nodes. These replication requests that are concurrent will then be processed sequentially on the basis of the ascending order of their respective access time. In case the replication request i poses a greater QoS requirement in comparison to the replication request j, then the replication request i is then associated with a lesser access time compared to the replication request j. In a case such as this, the HQFR algorithm will at first begin the processing of the replication request i for storing its respective data replicas. During the processing of a QoS-aware replication request made from the node r_i, it is necessary that the relatively qualified nodes satisfying the QoS requirement of the running

application in r_i has to be found. It is generally known that the access time for a data block is useful for representing the QoS requirement corresponding to a data-intensive application. It is considered that the QoS requirement related to the running application in r_i is actually $T_{qos}(r_i)$ time units. In case the node q_j would want to be a qualified node of r_i, it is required to satisfy the two conditions that follow.

The nodes q_j and r_i cannot be placed in the same rack. This condition is in consideration of the possible failure of rack

$$R(r_i) \neq (q_j) \tag{5}$$

where R refers to the function for determining the rack where a node is placed.

The data replica access time taken from q_j to r_i ($T_{access}(r_i; q_j)$ is required to satisfy the $T_{qos}(r_i)$ constraint

$$T_{access(r_i,q_j)} = T_{disk}(q_j) + T_{comm}(r_i, q_j) \leq T_{qos}(r_i) \tag{6}$$

where $T_{disk}(q_j)$ refers to the disk access latency for the retrieval of a data block replica out from the disk of qj, and $T_{comm}(r_i; q_j)$ stands for the network communication latency for the transmission of a data block replica from q_j to r_i

In accordance with the two conditions above, all of the nodes that are qualified corresponding to the requested node r_i can be discovered. Then, r_f qualified nodes are chosen from all of the qualified nodes. These r_f qualified nodes have data replica access time that are smaller compared to the other qualified nodes. Subsequently, the data block of r_i will be, correspondingly, contain one replica to be stored in every r_f qualified nodes. These r_f qualified nodes will in addition. update their available replication space, respectively. The cost of replication is expressed as the summation of the storage expenses of all of the data block replicas, as shown below:

$$\sum_{\forall i \in SN_r} \sum_{\forall q_j \in SN_n^{r_i}} T_{storage}(r_i, q_j) \tag{7}$$

$T_{storage}(r_i, q_j)$ is similar to $T_{access}(r_i,q_j)$. In (6), have defined clearly $T_{access}(r_i, q_j)$ to be the addition of the network communication latency and the disk access latency for the retrieval of a data block replica from the node q_j to node r_i. Therefore, $T_{storage}(r_i, q_j)$ contains the time for transmitting a data block replica from r_i to node q_j and the time for writing the data block replica onto the disk of q_j. The optimal solution for the data replication issue can be got making use of Multi-Stage Stochastic Integer Programming (MSSIP). The MSSIP is a popular mechanism that is employed for solving the optimal problems having these characteristics: a linear objective function, several linear constraints, and a set of integer solution. Provided an example P of the data replication issue, the respective MSSIP can be formulated as (8) to (12). The notations used can be seen in Table 1. In the ILP formulation given, the placement of the data replica can be got on the basis of the binary variables x and y. If $x(r_i, q_j)$ is 1, the node q_j has one data block replica stored of the requested node r_i. If $y(r_i, q_j)$ is 1 too, the respective data block replica is a QoS-violated replica.

$$\text{Min} \quad \left(\sum_{\forall i \in SN_r} \sum_{\forall q_j \in SN_n^{\overline{R(r_i)}}} x(r_i, q_j) \times T_{storage(r_i,q_j)} \times \alpha(r_i, q_j) + y(r_i, q_j) \times T_{storage(r_i,q_j)} \times k(r_i, q_j) \right) \tag{8}$$

Subject to ,

$$\forall q_j \in SN \ , \ \sum_{\forall i \in SN_r} x(r_i, q_j) \leq a(q_j) \tag{9}$$

$$\forall r_j \in SN_r \ \wedge \ \forall q_j \in \overline{S_q^{r_i}}, y(r_i, q_j) = x(r_i, q_j) \tag{10}$$

$$\forall r_j \in SN_r \ \wedge \ \forall q_j \in S_q^{r_i}, y(r_i, q_j) = 0 \tag{11}$$

$$\forall r_j \in SN_r \ \wedge \ \forall q_j \in \overline{S_q^{r_i}}, x(r_i, q_j), y(r_i, q_j) \in \{0,1\} \tag{12}$$

$$\forall q_j \in SN_r, \sum_{\forall q_j \in SN_n^{\overline{R(r_i)}}} x(r_i, q_j) = \mathrm{r_f} \tag{13}$$

where $\alpha(r_i, q_j)$ and $k(r_i, q_j)$ are the constraint variable of unqualified nodes and coefficient k is made use for guaranteeing that the number of QoS-violated data replicas will first be the respective minimized parameters,. By summing up all the values of y, the total number of QoS-violated data replicas can be got. This number is anticipated to be as small as possible through the association with a fixed coefficient,

$$k = \max_{\forall r_i \in SN_r \wedge \forall q_j \in S} \{(T_{storage}(r_i, q_j)\} + 1 \tag{14}$$

Once the k is set, every $y(r_i, q_j)$ has a greater coefficient compared to every $x(r_i, q_j)$. It is also familiar that the values of $x(r_i, q_j)$and $y(r_i, q_j)$ are either 0 or 1. To extend the (8) to a random setting, consider that the replica problem parameters as $(\alpha(r_i, q_j),\ k(r_i, q_j),\ T_{Access}(r_i, q_j))$ evolve as replica access time from q_j to r_i having a finite probability space and generate a filtration. This information structure can be clarified as a scenario tree in which the storage nodes n in stage (or degree) t of the tree comprise the states belonging to the world which can be differentiated by the information that is available till time stage t with r_f. The probability that is associated with the state the node n is in p_n.

$$\text{Min} \qquad \text{Min}\begin{pmatrix} \sum_{\forall i \in SN_r} \mathrm{p_n} \sum_{\forall q_j \in SN_n^{\overline{R(r_i)}}} x(r_i, q_j) \times T_{storage(r_i, q_j)} \times \alpha(r_i, q_j) \\ +y(r_i, q_j) \times T_{storage(r_i, q_j)} \times k(r_i, q_j) \end{pmatrix} \tag{15}$$

$$\forall q_j \in SN_r, \sum_{m \in p_n} \sum_{\forall q_j \in SN_n^{\overline{R(r_i)}}} x_m(r_i, q_j) = \mathrm{r_f} \tag{16}$$

$T_{Access}(r_i, q_j)$ denotes the replica access time from q_j to r_i, and $T_{storage}(r_i, q_j)$ the storage time to store one data replica from the r_i to q_j. The summation in constraint (9) can be changed to

$$\sum_{m \in p_n} p(n) \backslash T_{Access}(r_i, q_j) \tag{17}$$

With the management method above, the nodes having similar CPU and disk performance are generally located in the same rack. These nodes are also in connection with one another making use of similar bandwidth through the switch of the rack. On the basis of this manner of node deployment, in case two or more number of requested nodes are placed in the same rack, then they are appropriate to be merged to form a requested rack node. In a similar way, the qualified nodes present in the same rack are also merged to form a qualified rack node. Along with combining the requested and qualified nodes on the basis of the rack unit, the below equations are employed for modeling the relationships existing between the requested rack nodes and the qualified rack nodes over the reduced flow graph. For the case of a requested rack node $\mathrm{rr_m}$, the total number of requests for replication is

$$|S\_rr_m| \tag{18}$$

where S_rr_m refers to the set of requested nodes in $\mathrm{rr_m}$. Every requested node makes one replication request at every time. The cardinality of S_rr_m can be useful for representing the total number of replication requests made in $\mathrm{rr_m}$. The QoS requirement corresponding to $\mathrm{rr_m}$ is given by

$$\min_{\forall r_i \in S\_rr_M} T_{qos}(r_i) \tag{19}$$

where $\mathrm{T_{qos}(r_i)}$ has been considered to be the QoS requirement pertaining to the request node $\mathrm{r_i}$. For all of the requested nodes in rrm, they can have diverse QoS requirements. It has been observed that the QoS requirement is denoted as the access time desired for a data block. In (11), the smallest access time is employed for representing the QoS requirement of $\mathrm{rr_m}$. This is for finding the respective qualified nodes of $\mathrm{rr_m}$ in a convenient manner. When the qualified node $\mathrm{q_j}$ can satisfy the QoS requirement of $\mathrm{rr_m}$, it should

meet the QoS requirements of all of the requested nodes in rr_m. For a qualified rack node qr_n, the size of its replication space equals to the sum of the replication space of every qualified node in qr_n, as expressed below,

$$\sum_{\forall q_j \in S\_qr_n} a(q_j), \tag{20}$$

where S_{qrn} and $a(q_j)$ are considered to be the set having the qualified nodes in qr_n and the available replication space of a qualified node q_j, correspondingly.

$$\sum_{\forall r_i S\_rr_m \wedge q_j \in S\_qr_n} T_{storage}(r_i, q_j), \tag{21}$$

where $T_{storage}(r_i; q_j)$ is defined as the storage cost incurred in storing a data replica from r_i to q_j. There are diverse respective requested nodes and qualified nodes in rr_m and qr_n.

4. PERFORMANCE EVALUATION

In the experimentation work performed simulations using a distributed cloud model that is based on the P2P overlay Kademlia [24]. Implemented proposed algorithm for secret sharing on the distributed cloud. Assumed that resource providers have a minimum of 2GB RAM up to 16 GB, 2 to 8 cores. First a node identifies available resource providers near him and then divides the key into multiple shares and distributes each share to an available resource provider. The Resource provider again creates more shares by using his share as the secret and stores it. User will maintain the list of providers who store the secret shares. When a user requires the key, he contacts other resource providers who have the shares. Once resource providers receive the request, they combine the shares they have and send the actual share to the user.

Figure 3 illustrates the average total time of the Secret Sharing (SS), Multilevel Threshold Secret Sharing (MTSS) and the proposed Multilevel Differential Evolution Threshold (MDET) –SS schema. The total time includes time taken to split the secret into shares at first level, find resources, distribute the shares to resource providers and split the shares at second level. It can see that as the number of nodes increases, the time to find nodes and distribute shares to nearby nodes decreases. Since the distributed cloud is formed by many users for secret sharing is feasible and efficient.

Figure 4 shows the total replication costs incurred for multiple numbers of requested nodes ranging from 500 to 2,500. In Figure. 4 the cloud computing system is configured with device heterogeneity making use of the first three disk access time and transmission rates tabulated in Table 2. The replication factor r_f is fixed at 2. Actually, the Hadoop replication algorithm follows a random mechanism to locate the replicas corresponding to a data block, but it also takes the possible rack failure into consideration. This way, the total replication expense of the Hadoop replication algorithm shows similarity with that of the random replication algorithm. Both the algorithms do not take care of the applications' QoS requirements in data replication. The total replication cost incurred for the proposed MSSIP is less when compared to other HDFS and random replication algorithm.

Figure 5 illustrates the comparison made for the average recovery time for the case of a corrupted data block. In case, the requested node r_i is not able to read a data block from its disk because of data corruption, how long it takes for r_i to have one replica of the data bock retrieved from another node.

The QoS violation ratio is defined as below:

= The total number of QoS - violated data block replicas/ The total number of data block replicas (22)

Figure 6 indicates the comparison made for the QoS violation ratios in the above mentioned algorithms. In Figure 6, the QoS violation ratios of these two abovementioned algorithms are nearly 59 and 57 percent,

correspondingly. The QoS requirement is taken into consideration in the replication algorithms proposed. The QoS violated data replicas are produced owing to the lesser replication space in a node.

Along with the minimization of the replication cost, the MSSIP algorithm can also reduce the number of QoS-violated data replicas.

5. CONCLUSIONS AND FUTURE WORK

Cloud Computing is a remarkably new idea which has a good number of advantages for the benefit of its users; But, it also poses few security issues that may reduce its usage. The combination of multi-clouds and secret sharing algorithm is promising, but as of yet it deals with many uncertainties. The current work ensures implementation of Multilevel secret sharing scheme for distributed cloud computing. Multilevel Differential Evolution Threshold (MDET) based secret sharing scheme of security in the case of cloud computing, giving an overview of the current status of security in this emerging technology. The main ides behind the MDET secret sharing schemes, information is split into multiple shares and these shares are distributed among multiple users to ensure the security of secret key in the distributed cloud environment. In addition the proposed idea examined about the QoS-aware data replication issue in the field of distributed cloud computing. To solve the problem of data replication, the device heterogeneity is also taken into account along with the QoS requirements of applications. Multi-Stage Stochastic Integer Programming (MSSIP) is adopts the intuitive idea of Quality of Service (QoS) to perform data replication. In order to have the proposed MSSIP replication algorithm to be adaptive to a distributed cloud computing system, the node combination methods are also presented in order to deal with the scalable replication problem. An experimentation result shows that the proposed MDET based secret sharing scheme, as well as to identify relate vulnerabilities and threats with possible solutions when compared to existing state of art methods. With a futuristic approach, plans are in place to realize the new MSSIP algorithms in an original cloud computing platform. Furthermore, the replication algorithms will then also be extended to be involved in energy consumption. The maximum size of data is again one of the factors that have been also considered to deal with in practical implementation in future work.

REFERENCES

[1] Gartner Inc Gartner identifies the Top 10 strategic technologies for 2011. Online. Available: http://www.gartner.com/it/page.jsp?id=1454221. Accessed: 15-Jul-2011

[2] S. Zhang, S.Zhang, X. Chen, X. Huo, “Cloud Computing Research and Development Trend.’ In: Second International Conference on Future Networks (ICFN’10), Sanya, Hainan, China. IEEE Computer Society, Washington, DC, USA, pp 93–97, 2010.

[3] A.Khalid, “Cloud Computing: applying issues in Small Business.’ In: International Conference on Signal Acquisition and Processing (ICSAP’10), pp. 278–281, 2010.

[4] Khethavath, Praveen, J.Thomas, E.Chan-Tin, and H. Liu, “Introducing a distributed cloud architecture with efficient resource discovery and optimal resource allocation,” IEEE Ninth World Congress on In Services (SERVICES). pp. 386-392, 2013.

[5] D.P.Anderson, “Boinc: a system for public-resource computing and storage,” In: Proceedings of Fifth IEEE/ACM International Workshop on Grid Computing, 2004.

[6] Chun, Brent, D.Culler, T. Roscoe, A. Bavier, L.Peterson, M.Wawrzoniak, and M.Bowman. “Planetlab: an overlay testbed for broad-coverage services,” ACM SIGCOMM Computer Communication Review. Vol.33. No. 3. pp.3-12, 2003.

[7] M. Armbrust, A .Fox, R. Griffith, A.D. Joseph, R.H.Katz, A .Konwinski, G.Lee, D.A. Patterson, A. Rabkin, I.Stoica, and M. Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”. Technical Report UCB/EECS-2009-28, University of California at Berkeley, eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html, 2009.

[8] D.Catteddu, and G.Hogben, “Benefits, risks and recommendations for information security”, Tech. rep., European Network and Information Security Agency, enisa.europa.eu/act/rm/files/deliverables/cloudcomputing- risk-assessment, 2009.

[9] T. Mather, and S. Kumaraswamy, Cloud Security and privacy: An Enterprise Perspective on Risks and Compliance. 1st edition. O’Reilly Media. 2009.

[10] C.N. Reddy, "A CIM (Common Information Model) Based Management Model for Clouds.' Proc. IEEE Int'l Conf. Cloud Computing in Emerging Markets (CCEM), pp. 1-5, 2012.

[11] M. A. AlZain, E. Pardede, B. Soh, J. A, "Thom Cloud Computing Security: From Single to Multi-Clouds.' IEEE 45th Hawaii International Conference on System Sciences, 2012.

[12] M.Singhal, S.Chandrasekhar, TingjianGe., R.Sandhu, R. Krishnan, Gail-JoonAhn., E. Bertino, "Collaboration in Multicloud Computing Environments: Framework and Security Issues", IEEE computer society journal. Vol. 46. No. 2. pp. 76-84, 2013.

[13] J.Bohli, N.Gruschka, M.Jensen, L.Lo Iacono, N.Marnau, "Security and Privacy Enhancing Multi-Cloud Architectures", IEEE Transaction on Dependable and secure computing . No.99, 2013.

[14] S.Takahashi, and K. Iwamura, "Secret Sharing Scheme Suitable for Cloud Computing.' In IEEE 27th International Conference on Advanced Information Networking and Applications (AINA), pp. 530-537, 2013.

[15] S.Chen, Y.Chen, H.Jiang, L. T. Yang, and K. C. Li, "A secure distributed file system based on revised Blakley's secret sharing scheme.' 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 310-317, 2012.

[16] F.Alsolami, and C.E. Chow, "N-Cloud: improving performance and security in cloud storage.' In: IEEE 14th International Conference on High Performance Switching and Routing (HPSR), pp. 221 – 222, 2013.

[17] H.Xiong, X.Zhang, W. Zhu, and D. Yao, Cloudseal: End-to-end content protection in cloud-based storage and delivery services. In Security and Privacy in Communication Networks. Springer Berlin Heidelberg. pp. 491-500. 2012.

[18] Endo, P.Takako, A.Vitor de Almeida Palhares, N.N. Pereira, G. E. Goncalves, D.Sadok, J. Kelner, B. Melander, and J-E.Mångs. "Resource allocation for distributed cloud: concepts and research challenges", IEEE Network. Vol.25. No. 4. pp. 42-46, 2011.

[19] A. K.Qin, V. L.Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization", IEEE Transactions on Evolutionary Computation. Vol. 13. No.2. pp.398-417, 2009.

[20] A.Shamekhi, "An Improved Differential Evolution Optimization Algorithm", International Journal of Research and Reviews in Applied Sciences. Vol.15. No.2. pp.132-145, 2013.

[21] T. Chen, "Multilevel Threshold Secret Sharing in Distributed Cloud.' In Security in Computing and Communications: Third International Symposium, Kochi, India, pp.10-13, 2015.

[22] S.Ahmed, A. J.King, and G. Parija, "A multi-stage stochastic integer programming approach for capacity expansion under uncertainty", Journal of Global Optimization.Vol.26. No. 1. pp. 3-24, 2003.

[23] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," Proc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), pp. 1-10, 2010.

[24] P.Maymounkov, and D.Mazi'eres, "Kademlia: a peer-to-peer information system based on the XOR metric", In: Druschel, P., Kaashoek, M.F., Rowstron, A. (eds.) IPTPS 2002. LNCS, Vol. 2429, pp. 53–65. Springer, Heidelberg (2002).