# DROPS: Division and Replication of Data in Cloud for Optimal Performance and Security

Mazhar Ali, *Student Member, IEEE,* Kashif Bilal, *Student Member, IEEE,* Samee U. Khan, *Senior Member, IEEE,* Bharadwaj Veeravalli, *Senior Member, IEEE,* Keqin Li, *Senior Member, IEEE,* and Albert Y. Zomaya, *Fellow, IEEE*

**Abstract**—Outsourcing data to a third-party administrative control, as is done in cloud computing, gives rise to security concerns. The data compromise may occur due to attacks by other users and nodes within the cloud. Therefore, high security measures are required to protect data within the cloud. However, the employed security strategy must also take into account the optimization of the data retrieval time. In this paper, we propose Division and Replication of Data in the Cloud for Optimal Performance and Security (DROPS) that collectively approaches the security and performance issues. In the DROPS methodology, we divide a file into fragments, and replicate the fragmented data over the cloud nodes. Each of the nodes stores only a single fragment of a particular data file that ensures that even in case of a successful attack, no meaningful information is revealed to the attacker. Moreover, the nodes storing the fragments, are separated with certain distance by means of graph T-coloring to prohibit an attacker of guessing the locations of the fragments. Furthermore, the DROPS methodology does not rely on the traditional cryptographic techniques for the data security; thereby relieving the system of computationally expensive methodologies. We show that the probability to locate and compromise all of the nodes storing the fragments of a single file is extremely low. We also compare the performance of the DROPS methodology with ten other schemes. The higher level of security with slight performance overhead was observed.

**Index Terms**—Centrality, cloud security, fragmentation, replication, performance.

## 1 Introduction

The cloud computing paradigm has reformed the usage and management of the information technology infrastructure [7]. Cloud computing is characterized by on-demand self-services, ubiquitous network accesses, resource pooling, elasticity, and measured services [22, 8]. The aforementioned characteristics of cloud computing make it a striking candidate for businesses, organizations, and individual users for adoption [25]. However, the benefits of low-cost, negligible management (from a users perspective), and greater flexibility come with increased security concerns [7].

Security is one of the most crucial aspects among those prohibiting the wide-spread adoption of cloud computing [14, 19]. Cloud security issues may stem due to the core technology's implementation (virtual machine (VM) escape, session riding, etc.), cloud service offerings (structured query language injection, weak authentication schemes, etc.), and arising from cloud characteristics (data recovery vulnerability, Internet protocol vulnerability, etc.) [5]. For a cloud to be secure, all of the participating entities must be secure. In any given system with multiple units, the highest level of the system's security is equal to the security level of the weakest entity [12]. Therefore, in a cloud, the security of the assets does not solely depend on an individual's security measures [5]. The neighboring entities may provide an opportunity to an attacker to bypass the users defenses.

The off-site data storage cloud utility requires users to move data in cloud's virtualized and shared environment that may result in various security concerns. Pooling and elasticity of a cloud, allows the physical resources to be shared among many users [22]. Moreover, the shared resources may be reassigned to other users at some instance of time that may result in data compromise through data recovery methodologies [22]. Furthermore, a multi-tenant virtualized environment may result in a VM to escape the bounds of virtual machine monitor (VMM). The escaped VM can interfere with other VMs to have access to unauthorized data [9]. Similarly, cross-tenant virtualized network access may also compromise data privacy and integrity. Improper media sanitization can also leak customer's private data [5].

- *M. Ali, K. Bilal, and S. U. Khan are with the Department of Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58108-6050, USA. E-mail: {mazhar.ali,kashif.bilal,samee.khan}@ndsu.edu*
- *B. Veeravallii is with the Department of Electrical and Computer Engineering, The National University of Singapore. E-mail: elebv@nus.edu.sg*
- *K. Li is with the Department of Computer Science, State University of New York , New Paltz, NY 12561. E-mail: lik@ndsu.edu*
- *A.Y. Zomaya is with the School of Information Technologies, The University of Sydney, Sydney, NSW 2006, Australia. E-mail: albert.zomaya@sydney.edu.au*

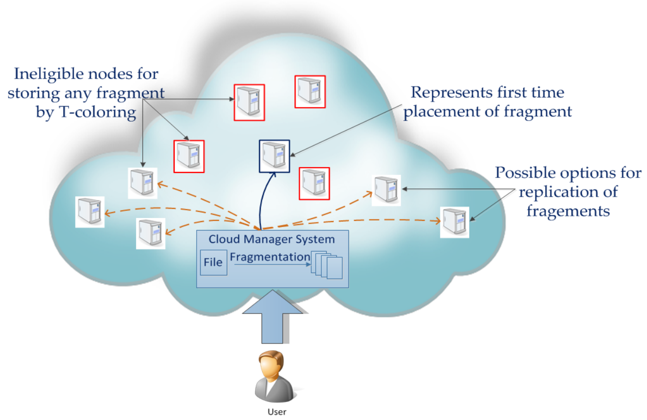

Ineligible nodes for storing any fragment by T-coloring
Represents first time placement of fragment
Possible options for replication of fragements
Cloud Manager System
File
Fragmentation
User

Fig. 1: The DROPS methodology

The data outsourced to a public cloud must be secured. Unauthorized data access by other users and processes (whether accidental or deliberate) must be prevented [14]. As discussed above, any weak entity can put the whole cloud at risk. In such a scenario, the security mechanism must substantially increase an attacker's effort to retrieve a reasonable amount of data even after a successful intrusion in the cloud. Moreover, the probable amount of loss (as a result of data leakage) must also be minimized.

A cloud must ensure throughput, reliability, and security [15]. A key factor determining the throughput of a cloud that stores data is the data retrieval time [21]. In large-scale systems, the problems of data reliability, data availability, and response time are dealt with data replication strategies [3]. However, placing replicas data over a number of nodes increases the attack surface for that particular data. For instance, storing $m$ replicas of a file in a cloud instead of one replica increases the probability of a node holding file to be chosen as attack victim, from $\frac{1}{n}$ to $\frac{m}{n}$, where $n$ is the total number of nodes.

From the above discussion, we can deduce that both security and performance are critical for the next generation large-scale systems, such as clouds. Therefore, in this paper, we collectively approach the issue of security and performance as a secure data replication problem. We present Division and Replication of Data in the Cloud for Optimal Performance and Security (DROPS) that judicially fragments user files into pieces and replicates them at strategic locations within the cloud. The division of a file into fragments is performed based on a given user criteria such that the individual fragments do not contain any meaningful information. Each of the cloud nodes (we use the term node to represent computing, storage, physical, and virtual machines) contains a distinct fragment to increase the data security. A successful attack on a single node must not reveal the locations of other fragments within the cloud. To keep an attacker uncertain about the locations of the file fragments and to further improve the security, we select the nodes in a manner that they are not adjacent and are at certain distance from each other. The node separation is ensured by the means of the T-coloring [6]. To improve data retrieval time, the nodes are selected based on the centrality measures that ensure an improved access time. To further improve the retrieval time, we judicially replicate fragments over the nodes that generate the highest read/write requests. The selection of the nodes is performed in two phases. In the first phase, the nodes are selected for the initial placement of the fragments based on the centrality measures. In the second phase, the nodes are selected for replication. The working of the DROPS methodology is shown as a high-level work flow in Fig. 1. We implement ten heuristics based replication strategies as comparative techniques to the DROPS methodology. The implemented replication strategies are: **(a)** A-star based searching technique for data replication problem (DRPA-star), **(b)** weighted A-star (WA-star), **(c)** A$\epsilon$-star, **(d)** suboptimal A-star1 (SA1), **(e)** suboptimal A-star2 (SA2), **(f)** suboptimal A-star3 (SA3), **(g)** Local Min-Min, **(h)** Global Min-Min, **(i)** Greedy algorithm, and **(j)** Genetic Replication Algorithm (GRA). The aforesaid strategies are fine-grained replication techniques that determine the number and locations of the replicas for improved system performance. For our studies, we use three Data Center Network (DCN) architectures, namely: **(a)** Three tier, **(b)** Fat tree, and **(c)** DCell. We use the aforesaid architectures because they constitute the modern cloud infrastructures and the DROPS methodology is proposed to work for the cloud computing paradigm.

Our major contributions in this paper are as follows:

- We develop a scheme for outsourced data that takes into account both the security and performance. The proposed scheme fragments and replicates the data file over cloud nodes.
- The proposed DROPS scheme ensures that even in the case of a successful attack, no meaningful information is revealed to the attacker.
- We do not rely on traditional cryptographic techniques for data security. The non-cryptographic nature of the proposed scheme makes it faster to perform the required operations (placement and retrieval) on the data.
- We ensure a controlled replication of the file fragments, where each of the fragments is replicated only once for the purpose of improved security.

The remainder of the paper is organized as follows. Section 2 provides an overview of the related work in the field. In Section 3, we present the preliminaries. The DROPS methodology is introduced in Section 4. Section 5 explains the experimental setup and results, and Section 6 concludes the paper.

## 2 Related Work

Juels *et al.* [10] presented a technique to ensure the integrity, freshness, and availability of data in a cloud. The data migration to the cloud is performed by the Iris file system. A gateway application is designed and employed in the organization that ensures the integrity and freshness of the data using a Merkle tree. The file blocks, MAC codes, and version numbers are stored at various levels of the tree. The proposed technique in [10] heavily depends on the user's employed scheme for data confidentiality. Moreover, the probable amount of loss in case of data tempering as a result of intrusion or access by other VMs cannot be decreased. Our proposed strategy does not depend on the traditional cryptographic techniques for data security. Moreover, the DROPS methodology does not store the whole file on a single node to avoid compromise of all of the data in case of successful attack on the node.

The authors in [11] approached the virtualized and multi-tenancy related issues in the cloud storage by utilizing the consolidated storage and native access control. The Dike authorization architecture is proposed that combines the native access control and the tenant name space isolation. The proposed system is designed and works for object based file systems. However, the leakage of critical information in case of improper sanitization and malicious VM is not handled. The DROPS methodology handles the leakage of critical information by fragmenting data file and using multiple nodes to store a single file.

The use of a trusted third party for providing security services in the cloud is advocated in [22]. The authors used the public key infrastructure (PKI) to enhance the level of trust in the authentication, integrity, and confidentiality of data and the communication between the involved parties. The keys are generated and managed by the certification authorities. At the user level, the use of temper proof devices, such as smart cards was proposed for the storage of the keys. Similarly, Tang *et. al.* have utilized the public key cryptography and trusted third party for providing data security in cloud environments [20]. However, the authors in [20] have not used the PKI infrastructure to reduce the overheads. The trusted third party is responsible for the generation and management of public/private keys. The trusted third party may be a single server or multiple servers. The symmetric keys are protected by combining the public key cryptography and the *(k, n)* threshold secret sharing schemes. Nevertheless, such schemes do not protect the data files against tempering and loss due to issues arising from virtualization and multi-tenancy.

A secure and optimal placement of data objects in a distributed system is presented in [21]. An encryption key is divided into $n$ shares and distributed on different sites within the network. The division of a key into $n$ shares is carried out through the *(k, n)* threshold secret sharing scheme. The network is divided into clusters. The number of replicas and their placement is determined through heuristics. A primary site is selected in each of the clusters that allocates the replicas within the cluster. The scheme presented in [21] combines the replication problem with security and access time improvement. Nevertheless, the scheme focuses only on the security of the encryption key. The data files are not fragmented and are handled as a single file. The DROPS methodology, on the other hand, fragments the file and store the fragments on multiple nodes. Moreover, the DROPS methodology focuses on the security of the data within the cloud computing domain that is not considered in [21].

## 3 Preliminaries

Before we go into the details of the DROPS methodology, we introduce the related concepts in the following for the ease of the readers.

### 3.1 Data Fragmentation

The security of a large-scale system, such as cloud depends on the security of the system as a whole and the security of individual nodes. A successful intrusion into a single node may have severe consequences, not only for data and applications on the victim node, but also for the other nodes. The data on the victim node may be revealed fully because of the presence of the whole file [17]. A successful intrusion may be a result of some software or administrative vulnerability [17]. In case of homogenous systems, the same flaw can be utilized to target other nodes within the system. The success of an attack on the subsequent nodes will require less effort as compared to the effort on the first node. Comparatively, more effort is required for heterogeneous systems. However, compromising a single file will require the effort to penetrate only a single node. The amount of compromised data can be reduced by making fragments of a data file and storing them on separate nodes [17, 21]. A successful intrusion on a single or few nodes will only provide access to a portion of data that might not be of any significance. Moreover, if an attacker is uncertain about the locations of the fragments, the probability of finding fragments on all of the nodes is very low. Let us consider a cloud with $M$ nodes and a file with $z$ number of fragments. Let $s$ be the number of successful intrusions on distinct nodes, such that $s>z$. The probability that $s$ number of victim nodes contain all of the $z$ sites storing the file fragments (represented by $P(s,z)$) is given as:

$$P(s,z) = \frac{\binom{s}{z}\binom{M-s}{s-z}}{\binom{M}{s}}. \tag{1}$$

If $M = 30$, $s = 10$, and $z = 7$, then $P(10, 7) = 0.0046$. However, if we choose $M = 50$, $s = 20$, and $z = 15$, then $P(20, 15) = 0.000046$. With the increase in $M$, the probability of a state reduces further. Therefore, we can say that the greater the value of $M$, the less probable that an attacker will obtain the data file. In cloud systems with thousands of nodes, the probability for an attacker to obtain a considerable amount of data, reduces significantly. However, placing each fragment once in the system will increase the data retrieval time. To improve the data retrieval time, fragments can be replicated in a manner that reduces retrieval time to an extent that does not increase the aforesaid probability.

## 3.2 Centrality

The centrality of a node in a graph provides the measure of the relative importance of a node in the network. The objective of improved retrieval time in replication makes the centrality measures more important. There are various centrality measures; for instance, closeness centrality, degree centrality, betweenness centrality, eccentricity centrality, and eigenvector centrality. We only elaborate on the closeness, betweenness, and eccentricity centralities because we are using the aforesaid three centralities in this work. For the remainder of the centralities, we encourage the readers to review [24].

### 3.2.1 Betweenness Centrality

The betweenness centrality of a node $n$ is the number of the shortest paths, between other nodes, passing through $n$ [24]. Formally, the betweenness centrality of any node $v$ in a network is given as:

$$C_b(v) = \sum_{a \neq v \neq b} \frac{\delta_{ab}(v)}{\delta ab}, \tag{2}$$

where $\delta_{ab}$ is the total number of shortest paths between $a$ and $b$, and $\delta_{ab}(v)$ is the number of shortest paths between $a$ and $b$ passing through $v$. The variable $C_b(v)$ denotes the betweenness centrality for node $v$.

### 3.2.2 Closeness Centrality

A node is said to be closer with respect to all of the other nodes within a network, if the sum of the distances from all of the other nodes is lower than the sum of the distances of other candidate nodes from all of the other nodes [24]. The lower the sum of distances from the other nodes, the more central is the node. Formally, the closeness centrality of a node $v$ in a network is defined as:

$$C_c(v) = \frac{N-1}{\sum_{a \neq v} d(v,a)}, \tag{3}$$

where $N$ is total number of nodes in a network and $d(v, a)$ represents the distance between node $v$ and node $a$.

TABLE 1: Notations and their meanings

| Symbols | Meanings |
|---|---|
| $M$ | Total number of nodes in the cloud |
| $N$ | Total number of file fragments to be placed |
| $O_k$ | $k$-th fragment of file |
| $o_k$ | Size of $O_k$ |
| $S^i$ | $i$-th node |
| $s_i$ | Size of $S^i$ |
| $cen_i$ | Centrality measure for $S^i$ |
| $col_{S^i}$ | Color assigned to $S^i$ |
| $T$ | A set containing distances by which assignment of fragments must be separated |
| $r_k^i$ | Number of reads for $O_k$ from $S^i$ |
| $R_k^i$ | Aggregate read cost of $r_k^i$ |
| $w_k^i$ | Number of writes for $O_k$ from $S^i$ |
| $W_k^i$ | Aggregate write cost of $w_k^i$ |
| $NN_k^i$ | Nearest neighbor of $S^i$ holding $O_k$ |
| c(i,j) | Communication cost between $S^i$ and $S^j$ |
| $P_k$ | Primary node for $O_k$ |
| $R_k$ | Replication schema of $O_k$ |
| RT | Replication time |

### 3.2.3 Eccentricity

The eccentricity of a node $n$ is the maximum distance to any node from a node $n$ [24]. A node is more central in the network, if it is less eccentric. Formally, the eccentricity can be given as:

$$E(v_a) = max_b d(v_a, v_b), \tag{4}$$

where $d(v_a, v_b)$ represents the distance between node $v_a$ and node $v_b$. It may be noted that in our evaluation of the strategies the centrality measures introduced above seem very meaningful and relevant than using simple hop-count kind of metrics.

## 3.3 T-coloring

Suppose we have a graph $G = (V, E)$ and a set $T$ containing non-negative integers including 0. The T-coloring is a mapping function $f$ from the vertices of $V$ to the set of non-negative integers, such that |f(x)- f(y)| $\notin T$, where $(x, y) \in E$. The mapping function $f$ assigns a color to a vertex. In simple words, the distance between the colors of the adjacent vertices must not belong to $T$. Formulated by Hale [6], the T-coloring problem for channel assignment assigns channels to the nodes, such that the channels are separated by a distance to avoid interference.

## 4 DROPS

### 4.1 System Model

Consider a cloud that consists of $M$ nodes, each with its own storage capacity. Let $S^i$ represents the name of $i$-th node and $s_i$ denotes total storage capacity of $S^i$. The communication time between $S^i$ and $S^j$ is the total time of all of the links within a selected path from $S^i$ to $S^j$ represented by *c(i, j)*. We consider $N$ number of file fragments such that $O_k$ denotes *k*-th fragment of a file while $o_k$ represents *the* size of *k-th* fragment. Let the total read and write requests from $S^i$ for $O_k$ be represented by $r_k^i$ and $w_k^i$, respectively. Let $P_k$ denote the primary node that stores the primary copy of $O_k$. The replication scheme for $O_k$ denoted by $R_k$ is also stored at $P_k$. Moreover, every $S^i$ contains a two-field record, storing $P_k$ for $O_k$ and $NN_k^i$ that represents the nearest node storing $O_k$. Whenever there is an update in $O_k$, the updated version is sent to $P_k$ that broadcasts the updated version to all of the nodes in $R_k$. Let *b(i,j)* and *t(i,j)* be the total bandwidth of the link and traffic between sites $S^i$ and $S^j$, respectively . The centrality measure for $S^i$ is represented by $cen_i$. Let $col_{S^i}$ store the value of assigned color to $S^i$. The $col_{S^i}$ can have one out of two values, namely: open_color and close_color. The value open_color represents that the node is available for storing the file fragment. The value close_color shows that the node cannot store the file fragment. Let $T$ be a set of integers starting from zero and ending on a prespecified number. If the selected number is three, then $T = \{0, 1, 2, 3\}$. The set $T$ is used to restrict the node selection to those nodes that are at hop-distances not belonging to $T$. For the ease of reading, the most commonly used notations are listed in Table 1.

Our aim is to minimize the overall total network transfer time or replication time (RT) or also termed as replication cost (RC). The RT is composed of two factors: **(a)** time due to read requests and **(b)** time due to write requests. The total read time of $O_k$ by $S^i$ from $NN_k^i$ is denoted by $R_k^i$ and is given by:

$$R_k^i = r_k^i o_k c(i, NN_k^i). \tag{5}$$

The total time due to the writing of $O_k$ by $S^i$ addressed to the $P_k$ is represented as $W_k^i$ and is given:

$$W_k^i = w_k^i o_k (c(i, P_k) + \sum_{(j \in R_k), j \neq i} c(P_k, j)). \tag{6}$$

The overall RT is represented by:

$$RT = \sum_{i=1}^{M} \sum_{k=1}^{N} (R_k^i + W_k^i) \tag{7}$$

The storage capacity constraint states that a file fragment can only be assigned to a node, if storage capacity of the node is greater or equal to the size of fragment. The bandwidth constraint states that $b(i,j) \geq t(i,j) \forall i,\ \forall j$. The DROPS methodology assigns the file fragments to the nodes in a cloud that minimizes the RT, subject to capacity and bandwidth constraints.

## 4.2 DROPS

In a cloud environment, a file in its totality, stored at a node leads to a single point of failure [17]. A successful attack on a node might put the data confidentiality or integrity, or both at risk. The aforesaid scenario can occur both in the case of intrusion or accidental errors. In such systems, performance in terms of retrieval time can be enhanced by employing replication strategies. However, replication increases the number of file copies within the cloud. Thereby, increasing the probability of the node holding the file to be a victim of attack as discussed in Section 1. Security and replication are essential for a large-scale system, such as cloud, as both are utilized to provide services to the end user. Security and replication must be balanced such that one service must not lower the service level of the other.

In the DROPS methodology, we propose not to store the entire file at a single node. The DROPS methodology fragments the file and makes use of the cloud for replication. The fragments are distributed such that no node in a cloud holds more than a single fragment, so that even a successful attack on the node leaks no significant information. The DROPS methodology uses controlled replication where each of the fragments is replicated only once in the cloud to improve the security. Although, the controlled replication does not improve the retrieval time to the level of full-scale replication, it significantly improves the security.

In the DROPS methodology, user sends the data file to cloud. The cloud manager system (a user facing server in the cloud that entertains user's requests) upon receiving the file performs: **(**a**)** fragmentation, **(**b**)** first cycle of nodes selection and stores one fragment over each of the selected node, and **(**c**)** second cycle of nodes selection for fragments replication. The cloud manager keeps record of the fragment placement and is assumed to be a secure entity.

The fragmentation threshold of the data file is specified to be generated by the file owner. The file owner can specify the fragmentation threshold in terms of either percentage or the number and size of different fragments. The percentage fragmentation threshold, for instance, can dictate that each fragment will be of 5% size of the total size of the file. Alternatively, the owner may generate a separate file containing information about the fragment number and size, for instance, fragment 1 of size 5,000 Bytes, fragment 2 of size 8,749 Bytes. We argue that the owner of the file is the best candidate to generate fragmentation threshold. The owner can best split the file such that each fragment does not contain significant amount of information as the owner is cognizant of all the facts pertaining to the data. The default percentage fragmentation threshold can be made a part of the Service Level Agreement (SLA), if the user does not specify the fragmentation threshold while uploading the data file. We primarily focus the storage system security in this work with an assumption that the communication channel between user and the cloud is secure.

**Algorithm 1** Algorithm for fragment placement

```
Inputs and initializations:
$O = \{O_1, O_2, ..., O_N\}$
$o = \{sizeof(O_1), sizeof(O_2), ...., sizeof(O_N)\}$
$col = \{open\_color, close\_color\}$
$cen = \{cen_1, cen_2, ..., cen_M\}$
$col \leftarrow open\_color \forall$ i
$cen \leftarrow cen_i \forall$ i
Compute:
for each $O_k \in O$ do
    select $S^i$ | $S^i \leftarrow$ indexof(max($cen_i$))
    if $col_{S^i}$ = $open\_color$ and $s_i >= o_k$ then
        $S^i \leftarrow O_k$
        $s_i \leftarrow s_i - o_k$
        $col_{S^i} \leftarrow close\_color$
        $S^{i\prime} \leftarrow distance(S^i, T)$    ▷ /*returns all nodes at distance $T$ from $S^i$ and stores in temporary set $S^{i\prime}$*/
        $col_{S^{i\prime}} \leftarrow close\_color$
    end if
end for
```

Once the file is split into fragments, the DROPS methodology selects the cloud nodes for fragment placement. The selection is made by keeping an equal focus on both security and performance in terms of the access time. We choose the nodes that are most central to the cloud network to provide better access time. For the aforesaid purpose, the DROPS methodology uses the concept of centrality to reduce access time. The centralities determine how central a node is based on different measures as discussed in Section 3.2. We implement DROPS with three centrality measures, namely: **(a)** betweenness, **(b)** closeness, and **(c)** eccentricity centrality. However, if all of the fragments are placed on the nodes based on the descending order of centrality, then there is a possibility that adjacent nodes are selected for fragment placement. Such a placement can provide clues to an attacker as to where other fragments might be present, reducing the security level of the data. To deal with the security aspects of placing fragments, we use the concept of T-coloring that was originally used for the channel assignment problem [6]. We generate a non-negative random number and build the set $T$ starting from zero to the generated random number. The set $T$ is used to restrict the node selection to those nodes that are at hop-distances not belonging to $T$. For the said purpose, we assign colors to the nodes, such that, initially, all of the nodes are given the open_color. Once a fragment is placed on the node, all of the nodes within the neighborhood at a distance belonging to $T$ are assigned close_color. In the aforesaid process, we lose some of the central nodes that may increase the retrieval time but we achieve a higher security level. If somehow the intruder compromises a node and obtains a fragment, then the location of the other fragments cannot be determined. The attacker can only keep on guessing the location of the other fragments. However, as stated previously in Section 3.1, the probability of a successful coordinated attack is extremely minute. The process is repeated until all of the fragments are placed at the nodes. Algorithm 1 represents the fragment placement methodology.

In addition to placing the fragments on the central nodes, we also perform a controlled replication to increase the data availability, reliability, and improve data retrieval time. We place the fragment on the node that provides the decreased access cost with an objective to improve retrieval time for accessing the fragments for reconstruction of original file. While replicating the fragment, the separation of fragments as explained in the placement technique through T-coloring, is also taken care off. In case of a large number of fragments or small number of nodes, it is also possible that some of the fragments are left without being replicated because of the T-coloring. As discussed previously, T-coloring prohibits to store the fragment in neighborhood of a node storing a fragment, resulting in the elimination of a number of nodes to be used for storage. In such a case, only for the remaining fragments, the nodes that are not holding any fragment are selected for storage randomly. The replication strategy is presented in Algorithm 2.

To handle the download request from user, the cloud manager collects all the fragments from the nodes and re-assemble them into a single file. Afterwards, the file is sent to the user.

**Algorithm 2** Algorithm for fragment's replication

```
for each $O_k$ in $O$ do
    select $S^i$ that has max($R_k^i + W_k^i$)
    if $col_{S^i}$ = open_color and $s_i >= o_k$ then
        $S^i \leftarrow O_k$
        $s_i \leftarrow s_i - o_k$
        $col_{S^i} \leftarrow close\_color$
        $S^{i\prime} \leftarrow distance(S^i, T)$    ▷ /*returns all nodes at distance $T$ from $S^i$ and stores in temporary set $S^{i\prime}$*/
        $col_{S^{i\prime}} \leftarrow close\_color$
    end if
end for
```

### 4.3 Discussion

A node is compromised with a certain amount of an attacker's effort. If the compromised node stores the data file in totality, then a successful attack on a cloud node will result in compromise of an entire data file. However, if the node stores only a fragment of a file, then a successful attack reveals only a fragment of a data file. Because the DROPS methodology stores fragments of data files over distinct nodes, an attacker has to compromise a large number of nodes to obtain meaningful information. The number of compromised

TABLE 2: Various attacks handled by DROPS methodology

| Attack | Description |
|---|---|
| Data Recovery | Rollback of VM to some previous state. May expose previously stored data. |
| Cross VM attack | Malicious VM attacking co-resident VM that may lead to data breach. |
| Improper media sanitization | Data exposure due to improper sanitization of storage devices. |
| E-discovery | Data exposure of one user due to seized hardware for investigations related to some other users. |
| VM escape | A malicious user or VM escapes from the control of VMM. Provides access to storage and compute devices. |
| VM rollback | Rollback of VM to some previous state. May expose previously stored data. |

nodes must be greater than $n$ because each of the compromised node may not give fragment in the DROPS methodology as the nodes are separated based on the T-coloring. Alternatively, an attacker has to compromise the authentication system of cloud [23]. The effort required by an attacker to compromise a node (in systems dealing with fragments/shares of data) is given in [23] as:

$$E_{Conf} = min(E_{Auth}, n \times E_{BreakIn}), \tag{8}$$

where $E_{Conf}$ is the effort required to compromise the confidentiality, $E_{Auth}$ is the effort required to compromise authentication, and $E_{BreakIn}$ is the effort required to compromise a single node. Our focus in this paper is on the security of the data in the cloud and we do not take into account the security of the authentication system. Therefore, we can say that to obtain $n$ fragments, the effort of an attacker increases by a factor of $n$. Moreover, in case of the DROPS methodology, the attacker must correctly guess the nodes storing fragments of file. Therefore, in the worst case scenario, the set of nodes compromised by the attacker will contain all of the nodes storing the file fragments. From Equation (1), we observe that the probability of the worst case to be successful is very low. The probability that some of the machines (average case) storing the file fragments will be selected is high in comparison to the worst case probability. However, the compromised fragments will not be enough to reconstruct the whole data. In terms of the probability, the worst, average, and best cases are dependent on the number of nodes storing fragments that are selected for an attack. Therefore, all of the three cases are captured by Equation (1).

Besides the general attack of a compromised node, the DROPS methodology can handle the attacks in which attacker gets hold of user data by avoiding or disrupting security defenses. Table 2 presents some of the attacks that are handled by the DROPS methodology. The presented attacks are cloud specific that stem from clouds core technologies. Table 2 also provides a brief description of the attacks. It is noteworthy that even in case of successful attacks (that are mentioned), the DROPS methodology ensures that the attacker gets only a fragment of file as DROPS methodology stores only a single fragment on the node. Moreover, the successful attack has to be on the node that stores the fragment.

## 5 Experimental Setup and Results

The communicational backbone of cloud computing is the Data Center Network (DCN) [2]. In this paper, we use three DCN architectures namely: **(a)** Three tier, **(b)** Fat tree, and **(c)** DCell [1]. The Three tier is the legacy DCN architecture. However, to meet the growing demands of the cloud computing, the Fat tree and Dcell architectures were proposed [2]. Therefore, we use the aforementioned three architectures to evaluate the performance of our scheme on legacy as well as state of the art architectures. The Fat tree and Three tier architectures are switch-centric networks. The nodes are connected with the access layer switches. Multiple access layer switches are connected using aggregate layer switches. Core layers switches interconnect the aggregate layer switches.. The Dcell is a server centric network architecture that uses servers in addition to switches to perform the communication process within the network [1]. A server in the Dcell architecture is connected to other servers and a switch. The lower level dcells recursively build the higher level dcells. The dcells at the same level are fully connected. For details about the aforesaid architectures and their performance analysis, the readers are encouraged to read [1] and [2].

### 5.1 Comparative techniques

We compared the results of the DROPS methodology with fine-grained replication strategies, namely: **(a)** DRPA-star, **(b)** WA-star, **(c)** A$\epsilon$-star, **(d)** SA1, **(e)** SA2, **(f)** SA3, **(g)** Local Min-Min, **(h)** Global Min-Min, **(i)** Greedy algorithm, and **(j)** Genetic Replication Algorithm (GRA). The DRPA-star is a data replication algorithm based on the A-star best-first search algorithm. The DRPA-star starts from the null solution that is called a root node. The communication cost at each node $n$ is computed as: $cost(n) = g(n) + h(n)$, where $g(n)$ is the path cost for reaching $n$ and $h(n)$ is called the heuristic cost and is the estimate of cost from $n$ to the goal node. The DRPA-star searches all of the solutions of allocating a fragment to a node. The solution that minimizes the cost within the constraints is explored while others are discarded. The selected solution is inserted into a list called the OPEN list. The list is ordered in the ascending order so that the solution with the minimum cost is expanded first. The heuristic used by the DRPA-star is given as $h(n) = max(0,(mmk(n)g(n)))$, where *mmk(n)* is the least cost replica allocation or the max-min RC. Readers are encouraged to see the details about DRPA-star in [13]. The WA-Star is a refinement of the DRPA-star that implements a weighted function to evaluate the cost. The function is given as: $f(n) = f(n) + h(n) + \epsilon(1 - (d(n)/D)h(n)$. The variable *d(n)* represents the depth of the node *n* and *D* denotes the expected depth of the goal node [13]. The A$\epsilon$-star is also a variation of the DRPA-star that uses two lists, OPEN and FOCAL. The FOCAL list contains only those nodes from the OPEN list that have $f$ greater than or equal to the lowest $f$ by a factor of $1 + \epsilon$. The node expansion is performed from the FOCAL list instead of the OPEN list. Further details about WA-Star and A$\epsilon$-star can be found in [13]. The SA1 (sub-optimal assignments), SA2, and SA3 are DRPA-star based heuristics. In SA1, at level *R* or below, only the best successors of node *n* having the least expansion cost are selected. The SA2 selects the best successors of node $n$ only for the first time when it reaches the depth level *R*. All other successors are discarded. The SA3 works similar to the SA2, except that the nodes are removed from OPEN list except the one with the lowest cost. Readers are encouraged to read [13] for further details about SA1, SA2, and SA3. The LMM can be considered as a special case of the bin packing algorithm. The LMM sorts the file fragments based on the RC of the fragments to be stored at a node. The LMM then assigns the fragments in the ascending order. In case of a tie, the file fragment with minimum size is selected for assignment (name local Min-Min is derived from such a policy). The GMM selects the file fragment with global minimum of all the RC associated with a file fragment. In case of a tie, the file fragment is selected at random. The Greedy algorithm first iterates through all of the $M$ cloud nodes to find the best node for allocating a file fragment. The node with the lowest replication cost is selected. The second node for the fragment is selected in the second iteration. However, in the second iteration that node is selected that produces the lowest RC in combination with node already selected. The process is repeated for all of the file fragments. Details of the greedy algorithm can be found in [18]. The GRA consists of chromosomes representing various schemes for storing file fragments over cloud nodes. Every chromosome consists of $M$ genes, each representing a node. Every gene is a $N$ bit string. If the *k*-th file fragment is to be assigned to $S^i$, then the *k*-th bit of *i-th* gene holds the value of one. Genetic algorithms perform the operations of selection, crossover, and mutation. The value for the crossover rate ($\mu_c$) was selected as $0.9$, while for the mutation rate ($\mu_m$) the value was $0.01$. The use of the values for $\mu_c$ and $\mu_m$ is advocated in [16].The best chromosome represents the solution. GRA utilizes mix and match strategy to reach the solution. More details about GRA can be obtained from [16].

## 5.2 Workload

The size of files were generated using a uniform distribution between 10Kb and 60 Kb. The primary nodes were randomly selected for replication algorithms. For the DROPS methodology, the $S^{i\prime}$s selected during the first cycle of the nodes selection by Algorithm 1 were considered as the primary nodes.

The capacity of a node was generated using a uniform distribution between $(\frac{1}{2}CS)C$ and $(\frac{3}{2}CS)C$, where $0 \leq C \geq 1$. For instance, for $CS = 150$ and $C = 0.6$ the capacities of the nodes were uniformly distributed between 45 and 135. The mean value of $g$ in the OPEN and FOCAL lists was selected as the value of $\epsilon$, for WA-star and A$\epsilon$-star, respectively. The value for level $R$ was set to $\lfloor \frac{d}{2} \rfloor$, where d is the depth of the search tree(number of fragments).

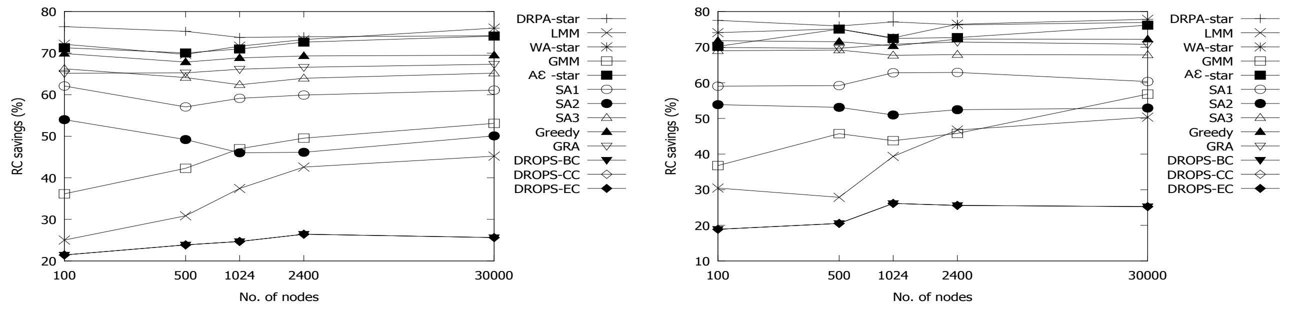
The read/write (R/W) ratio for the simulations that used fixed value was selected to be $0.25$ (The R/W ratio reflecting 25% reads and 75% writes within the cloud). The reason for choosing a high workload (lower percentage of reads and higher percentage of writes) was to evaluate the performance of the techniques under extreme cases. The simulations that studied the impact of change in the R/W ratio used various workloads in terms of R/W ratios. The R/W ratios selected were in the range of $0.10$ to $0.90$. The selected range covered the effect of high, medium, and low workloads with respect to the R/W ratio.

## 5.3 Results and Discussion

We compared the performance of the DROPS methodology with the algorithms discussed in Section 5.1. The behavior of the algorithms was studied by: **(a)** increasing the number of nodes in the system, **(b)** increasing the number of objects keeping number of nodes constant, **(c)** changing the nodes storage capacity, and **(d)** varying the read/write ratio. The aforesaid parameters are significant as they affect the problem size and the performance of algorithms [13].

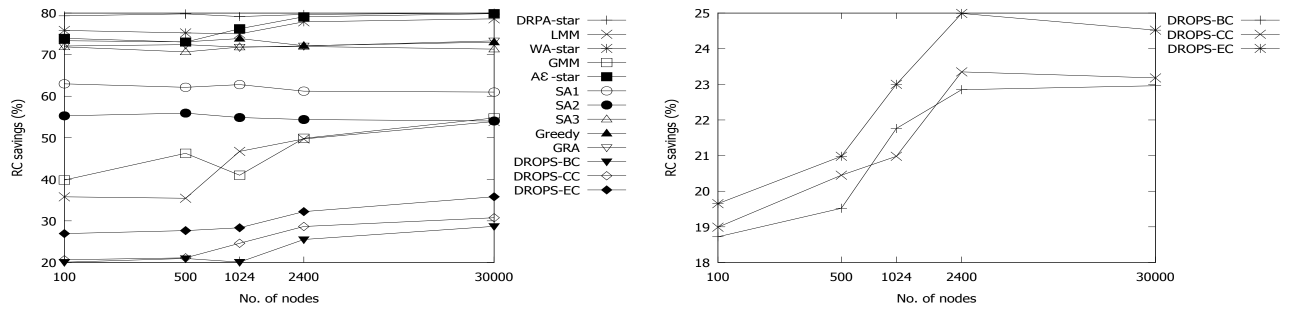
### *5.3.1 Impact of increase in number of cloud nodes*

We studied the performance of the placement techniques and the DROPS methodology by increasing the number of nodes. The performance was studied for the three discussed cloud architectures. The numbers of nodes selected for the simulations were 100, 500, 1,024, 2,400, and 30,000. The number of nodes in the Dcell architecture increases exponentially [2]. For a Dcell architecture, with two nodes in the $Dcell_0$, the architecture consists of 2,400 nodes. However, increasing a single node in the $Dcell_0$, the total nodes increases to 30, 000 [2]. The number of file fragments



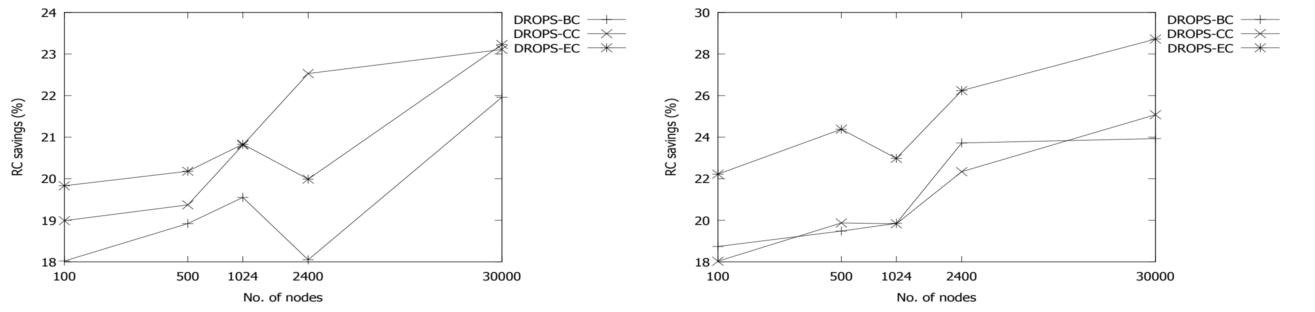
(a) (b)

Fig. 2: (a) RC versus number of nodes (Three tier) (b) RC versus number of nodes (Fat tier)



(a) (b)

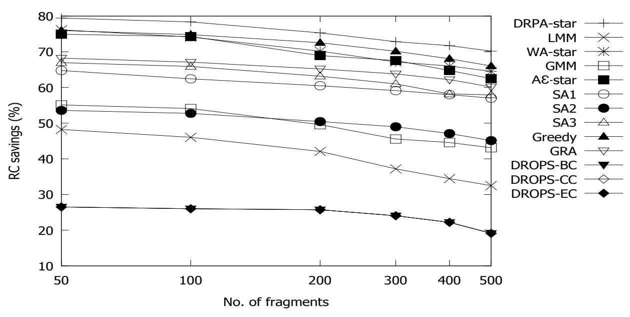
Fig. 3: (a) RC versus number of nodes (Dcell) (b) RC versus number of nodes for DROPS variations with maximum available capacity constraint (Three tier)



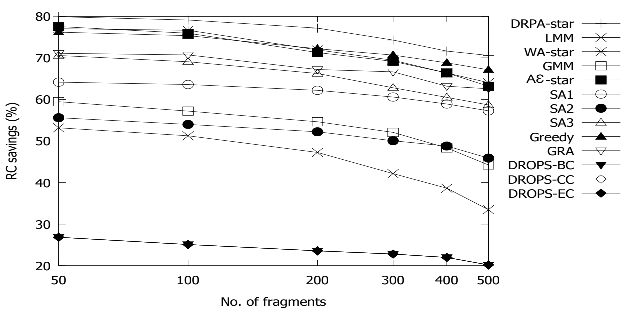
(a) (b)

Fig. 4: RC versus number of nodes for DROPS variations with maximum available capacity constraints (a) Fat tree (b) Dcell

was set to 50. For the first experiment we used $C = 0.2$. Fig. 2 (a), Fig. 2 (b), and Fig. 3 (a) show the results for the Three tier, Fat tree, and Dcell architectures, respectively. The reduction in network transfer time for a file is termed as *RC*. In the figures, the BC stands for the betweenness centrality, the CC stands for closeness centrality, and the EC stands for eccentricity centrality. The interesting observation is that although all of the algorithms showed similar trend in performance within a specific architecture, the performance of the algorithms was better in the Dcell architecture as compared to three tier and fat tree architectures. This is because the Dcell architecture exhibits better inter node connectivity and robustness [2]. The DRPA-star gave best solutions as compared to other techniques and registered consistent performance with the increase in the number of nodes. Similarly, WA-star, A$\epsilon$-star, GRA, greedy, and SA3 showed almost consistent performance with various number of nodes. The performance of LMM and GMM gradually increased with the increase in number of nodes since the increase in the number of nodes increased the number of bins. The SA1 and SA2 also showed almost constant performance in all of the three architectures. However, it is important to note that SA2 ended up with a decrease in performance



(a)



(b)

Fig. 5: (a) RC versus number of file fragments (Three tier) (b) RC versus number of file fragments (Fat tier)

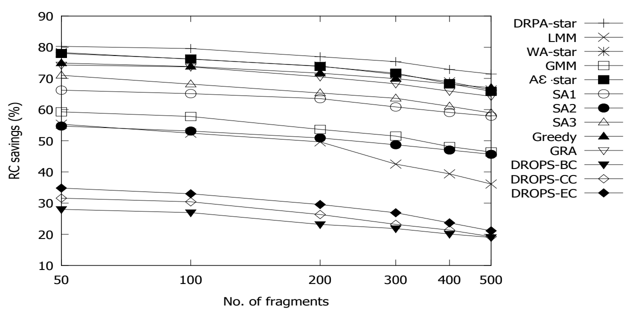
as compared to the initial performance. This may be due to the fact that SA2 only expands the node with minimum cost when it reaches at certain depth for the first time. Such a pruning for the first time, might have purged nodes by providing better global access time. The DROPS methodology, did not employ full-scale replication. Every fragment is replicated only once in the system. The smaller number of replicas of any fragment and separation of nodes by T-coloring decreased the probability of finding that fragment by an attacker. Therefore, the increase in the security level of the data is accompanied by the drop in performance as compared to the comparative techniques discussed in this paper. It is important to note that the DROPS methodology was implemented using three centrality measures namely: **(a)** betweenness, **(b)** closeness, and **(c)** eccentricity. However, Fig. 2(a) and Fig. 2(b) show only a single plot. Due to the inherent structure of the Three tier and Fat tree architectures, all of the nodes in the network are at the same distance from each other or exist at the same level. Therefore, the centrality measure is the same for all of the nodes. This results in the selection of same node for storing the file fragment. Consequently, the performance showed the same value and all three lines are on the same points. However, this is not the case for the Dcell architecture. In the Dcell architecture, nodes have different centrality measures resulting in the selection of different nodes. It is noteworthy to mention that in Fig 3(a), the eccentricity centrality performs better as compared to the closeness and betweenness centralities because the nodes with higher eccentricity are located closer to all other nodes within the network. To check the effect of closeness and betweenness centralities, we modified the heuristic presented in Algorithm 1. Instead of selecting the node with criteria of only maximum centrality, we selected the node with: **(a)** maximum centrality and **(b)** maximum available storage capacity. The results are presented in Fig. 3 (b), Fig. 4 (a), and Fig. 4 (b). It is evident that the eccentricity centrality resulted in the highest performance while the betweenness centrality showed the lowest performance. The reason for this is that nodes with higher eccentricity are closer to all other nodes in the network that results in lower RC value for accessing the fragments.

### 5.3.2 *Impact of increase in number of file fragments*

The increase in number of file fragments can strain the storage capacity of the cloud that, in turn may affect the selection of the nodes. To study the impact on performance due to increase in number of file fragments, we set the number of nodes to 30,000. The numbers of file fragments selected were 50, 100, 200, 300, 400, and 500. The workload was generated with $C = 45\%$ to observe the effect of increase number of file fragments with fairly reasonable amount of memory and to discern the performance of all the algorithms. The results are shown in Fig. 5 (a), Fig. 5 (b), and Fig. 6 (a) for the Three tier, Fat tree, and Dcell architectures, respectively. It can be observed from the plots that the increase in the number of file fragments reduced the performance of the algorithms, in general. However, the greedy algorithm showed the most improved performance. The LMM showed the highest loss in performance that is little above 16%. The loss in performance can be attributed to the storage capacity constraints that prohibited the placements of some fragments at nodes with optimal retrieval time. As discussed earlier, the DROPS methodology produced similar results in three tier and fat tree architectures. However, from the Dcell architecture, it is clear that the DROPS methodology with eccentricity centrality maintains the supremacy on the other two centralities.

### 5.3.3 *Impact of increase in storage capacity of nodes*

Next, we studied the effect of change in the nodes storage capacity. A change in storage capacity of the nodes may affect the number of replicas on the node due to storage capacity constraints. Intuitively, a lower node storage capacity may result in the elimination of some optimal nodes to be selected for replication because of violation of storage capacity constraints. The elimination of some nodes may degrade the performance to some extent because a node giving lower access time might be pruned due to non-availability



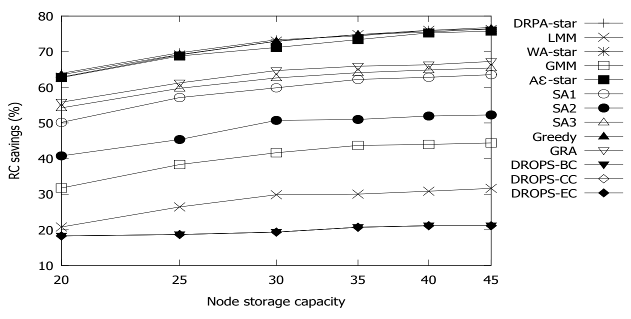
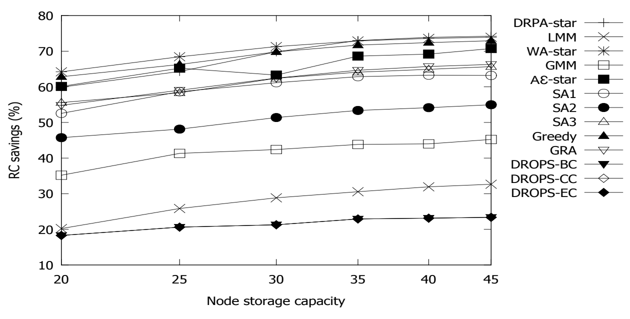


Fig. 6: (a) RC versus number of file fragments (Dcell) (b) RC versus nodes storage capacity (Three tier)



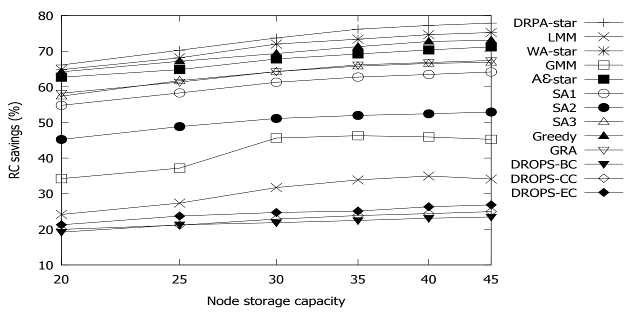
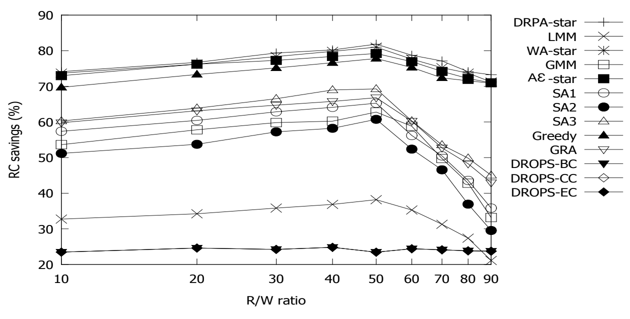


Fig. 7: (a) RC versus nodes storage capacity (Fat tree) (b) RC versus nodes storage capacity (Dcell)



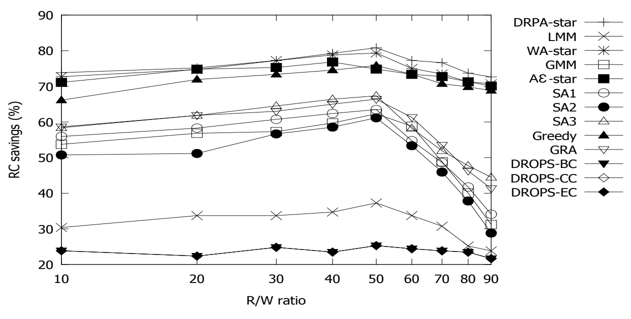


Fig. 8: (a) RC versus R/W ratio (Three tree) (b) RC versus R/W ratio (Fat tree)

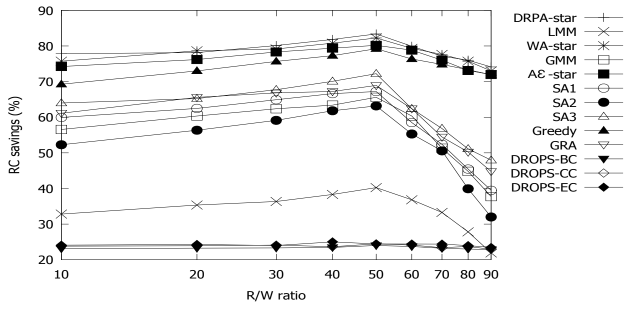


Fig. 9: RC versus R/W ratio (Dcell)

of enough storage space to store the file fragment. Higher node storage capacity allows full-scale replication of fragments, increasing the performance gain. However, node capacity above certain level will not change the performance significantly as replicating the already replicated fragments will not produce considerable performance increase. If the storage nodes have enough capacity to store the allocated file fragments, then a further increase in the storage capacity of a node cannot cause the fragments to be stored again. Moreover, the T-coloring allows only a single replica to be stored on any node. Therefore, after a certain point, the increase in storage capacity might not affect the performance.

We increase the nodes storage capacity incrementally from 20% to 40%. The results are shown in Fig. 6 (b), Fig. 7 (a), and Fig. 7 (b). It is observable from

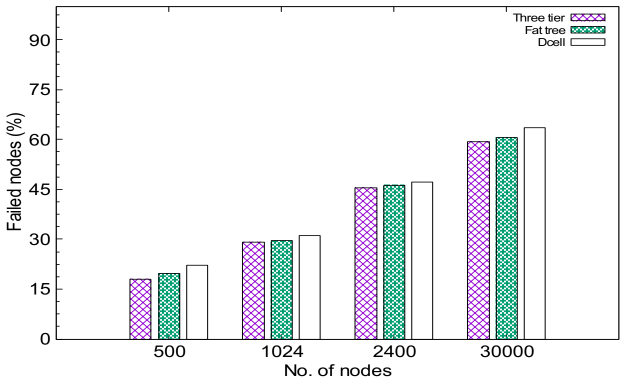


Fig. 10: Fault tolerance level of DROPS

the plots that initially, all of the algorithms showed significant increase in performance with an increase in the storage capacity. Afterwards, the marginal increase in the performance reduces with the increase in the storage capacity. The DRPA-star, greedy, WA-star, and A$\epsilon$-star showed nearly similar performance and recorded higher performance. The DROPS methodology did not show any considerable change in results when compared to previously discussed experiments (change in number of nodes and files). This is because the DROPS methodology does not go for a full-scale replication of file fragments rather they are replicated only once and a single node only stores a single fragment. Single time replication does not require high storage capacity. Therefore, the change in nodes storage capacity did not affect the performance of DROPS to a notable extent.

### 5.3.4 *Impact of increase in the read/write ratio*

The change in R/W ratio affects the performance of the discussed comparative techniques. An increase in the number of reads would lead to a need of more replicas of the fragments in the cloud. The increased number of replicas decreases the communication cost associated with the reading of fragments. However, the increased number of writes demands that the replicas be placed closer to the primary node. The presence of replicas closer to the primary node results in decreased RC associated with updating replicas. The higher write ratios may increase the traffic on the network for updating the replicas.

Fig. 8 (a), Fig. 8 (b), and Fig. 9 show the performance of the comparative techniques and the DROPS methodology under varying R/W ratios. It is observed that all of the comparative techniques showed an increase in the RC savings up to the R/W ratio of $0.50$. The decrease in the number of writes caused the reduction of cost associated with updating the replicas of the fragments. However, all of the comparative techniques showed some sort of decrease in RC saving for R/W ratios above $0.50$. This may be attributed to the fact that an increase in the number of reads caused more replicas of fragments resulting in increased cost of updating the replicas. Therefore, the increased cost of updating replicas underpins the advantage of decreased cost of reading with higher number of replicas at R/W ratio above $0.50$. It is also important to mention that even at higher R/W ratio values the DRPA-star, WA-star, A$\epsilon$-star, and Greedy algorithms almost maintained their initial RC saving values. The high performance of the aforesaid algorithms is due to the fact that these algorithms focus on the global RC value while replicating the fragments. Therefore, the global perception of these algorithms resulted in high performance. Alternatively, LMM and GMM did not show substantial performance due to their local RC view while assigning a fragment to a node. The SA1, SA2, and SA3 suffered due to their restricted search tree that probably ignored some globally high performing nodes during expansion. The DROPS methodology maintained almost consistent performance as is observable from the plots. The reason for this is that the DROPS methodology replicates the fragments only once, so varying R/W ratios did not affect the results considerably. However, the slight changes in the RC value are observed. This might be due to the reason that different nodes generate high cost for R/W of fragments with different R/W ratio.

As discussed earlier, the comparative techniques focus on the performance and try to reduce the RC as much as possible. The DROPS methodology, on the other hand, is proposed to collectively approach the security and performance. To increase the security level of the data, the DROPS methodology sacrifices the performance to certain extent. Therefore, we see a drop in the performance of the DROPS methodology as compared to discussed comparative techniques. However, the drop in performance is accompanied by much needed increase in security level.

Moreover, it is noteworthy that the difference in performance level of the DROPS methodology and the comparative techniques is least with the reduced storage capacity of the nodes (see Fig. 6 (b), Fig. 7 (a), and Fig. 7 (b)). The reduced storage capacity proscribes the comparative techniques to place as many replicas as required for the optimized performance. A further reduction in the storage capacity will tend to even lower the performance of the comparative techniques. Therefore, we conclude that the difference in performance level of the DROPS methodology and the comparative techniques is least when the comparative techniques reduce the extensiveness of replication for any reason.

Due to the fact that the DROPS methodology reduces the number of replicas, we have also investigates the fault tolerance of the DROPS methodology. If two nodes storing the same file fragment fail, the result will be incomplete or faulty file. We randomly picked and failed the nodes to check that what percentage of failed nodes will result in loss of data or

TABLE 3: Average RC (%) savings for increase in number of nodes

| Architecture | DRPA | LMM | wa-star | GMM | A$\epsilon$-star | SA1 | SA2 | SA3 | Greedy | GRA | DROPS-BC | DROPS-CC | DROPS-EC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Three tier | 74.70 | 36.23 | 72.55 | 45.62 | 71.82 | 59.86 | 49.09 | 64.38 | 69.1 | 66.1 | 24.41 | 24.41 | 24.41 |
| Fat tree | 76.76 | 38.95 | 75.22 | 45.77 | 73.33 | 60.89 | 52.67 | 68.33 | 71.64 | 70.54 | 23.28 | 23.28 | 23.28 |
| Dcell | 79.6 | 44.32 | 76.51 | 46.34 | 76.43 | 62.03 | 54.90 | 71.53 | 73.09 | 72.34 | 23.06 | 25.16 | 30.20 |

TABLE 4: Average RC (%) savings for increase in number of fragments

| Architecture | DRPA | LMM | wa-star | GMM | A$\epsilon$-star | SA1 | SA2 | SA3 | Greedy | GRA | DROPS-BC | DROPS-CC | DROPS-EC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Three tier | 74.63 | 40.08 | 69.69 | 48.67 | 68.82 | 60.29 | 49.65 | 62.18 | 71.25 | 64.44 | 23.93 | 23.93 | 23.93 |
| Fat tree | 75.45 | 44.33 | 70.90 | 52.66 | 70.58 | 61.12 | 51.09 | 64.64 | 71.73 | 66.90 | 23.42 | 23.42 | 23.42 |
| Dcell | 76.08 | 45.90 | 72.49 | 52.78 | 72.33 | 62.12 | 50.02 | 64.66 | 70.92 | 69.50 | 23.17 | 25.35 | 28.17 |

TABLE 5: Average RC (%) savings for increase in storage capacity

| Architecture | DRPA | LMM | wa-star | GMM | A$\epsilon$-star | SA1 | SA2 | SA3 | Greedy | GRA | DROPS-BC | DROPS-CC | DROPS-EC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Three tier | 72.37 | 28.26 | 71.99 | 40.63 | 71.19 | 59.29 | 48.67 | 61.83 | 72.09 | 63.54 | 19.89 | 19.89 | 19.89 |
| Fat tree | 69.19 | 28.34 | 70.73 | 41.99 | 66.20 | 60.28 | 51.29 | 61.83 | 69.33 | 62.16 | 21.60 | 21.60 | 21.60 |
| Dcell | 73.57 | 31.04 | 71.37 | 42.41 | 67.70 | 60.79 | 50.42 | 63.78 | 69.64 | 64.03 | 21.91 | 22.88 | 24.68 |

TABLE 6: Average RC (%) savings for increase in R/W ratio

| Architecture | DRPA | LMM | wa-star | GMM | A$\epsilon$-star | SA1 | SA2 | SA3 | Greedy | GRA | DROPS-BC | DROPS-CC | DROPS-EC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Three tier | 77.28 | 32.54 | 76.32 | 53.20 | 75.38 | 55.13 | 49.61 | 59.74 | 73.64 | 58.27 | 24.08 | 24.08 | 24.08 |
| Fat tree | 76.29 | 31.47 | 74.81 | 52.08 | 73.37 | 53.33 | 49.35 | 57.87 | 71.61 | 57.47 | 23.68 | 23.68 | 23.68 |
| Dcell | 78.72 | 33.66 | 78.03 | 55.82 | 76.47 | 57.44 | 52.28 | 61.94 | 74.54 | 60.16 | 23.32 | 23.79 | 24.23 |

selection of two nodes storing same file fragment. The numbers of nodes used in aforesaid experiment were 500, 1,024, 2,400, and 30, 000. The number of file fragments was set to 50. The results are shown in Fig. 10. As can be seen in Fig. 10, the increase in number of nodes increases the fault tolerance level. The random failure has generated a reasonable percentage for a soundly decent number of nodes.

We report the average RC (%) savings in Table 3, Table 4, Table 5, and Table 6. The averages are computed over all of the RC (%) savings within a certain class of experiments. Table 3 reveals the average results of all of the experiments conducted to observe the impact of increase in the number of nodes in the cloud for all of the three discussed cloud architectures. Table 4 depicts the average RC (%) savings for the increase in the number of fragments. Table 5 and Table 6 describe the average results for the increase the storage capacity and R/W ratio, respectively. It is evident from the average results that the Dcell architecture showed better results due to its higher connectivity ratio.

## 6 Conclusions

We proposed the DROPS methodology, a cloud storage security scheme that collectively deals with the security and performance in terms of retrieval time. The data file was fragmented and the fragments are dispersed over multiple nodes. The nodes were separated by means of T-coloring. The fragmentation and dispersal ensured that no significant information was obtainable by an adversary in case of a successful attack. No node in the cloud, stored more than a single fragment of the same file. The performance of the DROPS methodology was compared with full-scale replication techniques. The results of the simulations revealed that the simultaneous focus on the security and performance, resulted in increased security level of data accompanied by a slight performance drop.

Currently with the DROPS methodology, a user has to download the file, update the contents, and upload it again. It is strategic to develop an automatic update mechanism that can identify and update the required fragments only. The aforesaid future work will save the time and resources utilized in downloading, updating, and uploading the file again. Moreover, the implications of TCP incast over the DROPS methodology need to be studied that is relevant to distributed data storage and access.

## References

[1] K. Bilal, S. U. Khan, L. Zhang, H. Li, K. Hayat, S. A. Madani, N. Min-Allah, L. Wang, D. Chen, M. Iqbal, C. Z. Xu, and A. Y. Zomaya, "Quantitative comparisons of the state of the art data center architectures," *Concurrency and Computation: Practice and Experience*, Vol. 25, No. 12, 2013, pp. 1771-1783.

[2] K. Bilal, M. Manzano, S. U. Khan, E. Calle, K. Li, and A. Zomaya, "On the characterization of the structural robustness of data center networks," *IEEE Transactions on Cloud Computing*, Vol. 1, No. 1, 2013, pp. 64-77.

[3] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, "Energy-efficient data replication in cloud computing datacenters," *In IEEE Globecom Workshops*, 2013, pp. 446-451. .

[4] Y. Deswarte, L. Blain, and J-C. Fabre, "Intrusion tolerance in distributed computing systems," *In Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy, Oakland CA*, pp. 110-121, 1991.

[5] B. Grobauer, T.Walloschek, and E. Stocker, "Understanding cloud computing vulnerabilities," *IEEE Security and Privacy*, Vol. 9, No. 2, 2011, pp. 50-57.

[6] W. K. Hale, "Frequency assignment: Theory and applications," *Proceedings of the IEEE*, Vol. 68, No. 12, 1980, pp. 1497-1514.

[7] K. Hashizume, D. G. Rosado, E. Fernndez-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," *Journal of Internet Services and Applications*, Vol. 4, No. 1, 2013, pp. 1-13.

[8] M. Hogan, F. Liu, A.Sokol, and J. Tong, "NIST cloud computing standards roadmap," NIST Special Publication, July 2011.

[9] W. A. Jansen, "Cloud hooks: Security and privacy issues in cloud computing," *In 44th Hawaii IEEE International Conference onSystem Sciences (HICSS)*, 2011, pp. 1-10.

[10] A. Juels and A. Opera, "New approaches to security and availability for cloud data," *Communications of the ACM*, Vol. 56, No. 2, 2013, pp. 64-73.

[11] G. Kappes, A. Hatzieleftheriou, and S. V. Anastasiadis, "Dike: Virtualization-aware Access Control for Multitenant Filesystems," University of Ioannina, Greece, Technical Report No. DCS2013-1, 2013.

[12] L. M. Kaufman, "Data security in the world of cloud computing," *IEEE Security and Privacy*, Vol. 7, No. 4, 2009, pp. 61-64.

[13] S. U. Khan, and I. Ahmad, "Comparison and analysis of ten static heuristics-based Internet data replication techniques," *Journal of Parallel and Distributed Computing*, Vol. 68, No. 2, 2008, pp. 113-136.

[14] A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, "Towards Secure Mobile Cloud Computing: A Survey," *Future Generation Computer Systems*, Vol. 29, No. 5, 2013, pp. 1278-1299.

[15] A. N. Khan, M.L. M. Kiah, S. A. Madani, and M. Ali, "Enhanced dynamic credential generation scheme for protection of user identity in mobile-cloud computing, *The Journal of Supercomputing*, Vol. 66, No. 3, 2013, pp. 1687-1706 .

[16] T. Loukopoulos and I. Ahmad, "Static and adaptive distributed data replication using genetic algorithms," *Journal of Parallel and Distributed Computing*, Vol. 64, No. 11, 2004, pp. 1270-1285.

[17] A. Mei, L. V. Mancini, and S. Jajodia, "Secure dynamic fragment and replica allocation in large-scale distributed file systems," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 9, 2003, pp. 885-896.

[18] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," *In Proceedings of INFOCOM 2001, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, pp. 1587-1596, 2001.

[19] D. Sun, G. Chang, L. Sun, and X. Wang, "Surveying and analyzing security, privacy and trust issues in cloud computing environments," *Procedia Engineering*, Vol. 15, 2011, pp. 2852 2856.

[20] Y. Tang, P. P. Lee, J. C. S. Lui, and R. Perlman, "Secure overlay cloud storage with access control and assured deletion," *IEEE Transactions on Dependable and Secure Computing*, Vol. 9, No. 6, Nov. 2012, pp. 903-916.

[21] M. Tu, P. Li, Q. Ma, I-L. Yen, and F. B. Bastani, "On the optimal placement of secure data objects over Internet," *In Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium*, pp. 14-14, 2005.

[22] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, Vol. 28, No. 3, 2012, pp. 583-592.

[23] J. J. Wylie, M. Bakkaloglu, V. Pandurangan, M. W. Bigrigg, S. Oguz, K. Tew, C. Williams, G. R. Ganger, and P. K. Khosla, "Selecting the right data distribution scheme for a survivable storage system," Carnegie Mellon University, Technical Report CMU-CS-01-120, May 2001.

[24] M. Newman, *Networks: An introduction*, Oxford University Press, 2009.

[25] A. R. Khan, M. Othman, S. A. Madani, S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Communications Surveys and Tutorials*, DOI: 10.1109/SURV.2013.062613.00160.

**Mazhar Ali** is currently a PhD student at North Dakota State University, Fargo, ND, USA. His research interests include information security, formal verification, modeling, and cloud computing systems.

**Kashif Bilal** did his PhD in Electrical and Computer Engineering from the North Dakota State University, USA. His research interests include data center networks, distributed computing, and energy efficiency.

**Samee U. Khan** is an assistant professor at the North Dakota State University. His research interest include topics, such as sustainable computing, social networking, and reliability. He is a senior member of IEEE, and a fellow of IET and BCS.

**Bharadwaj Veeravalli** is an associate professor at the National University of Singapore. His main stream research interests include, Scheduling problems, Cloud/Cluster/Grid computing, Green Storage, and Multimedia computing. He is a senior member of the IEEE.

**Keqin Li** is a SUNY distinguished professor. His research interests include mainly in the areas of design and analysis of algorithms, parallel and distributed computing, and computer networking. He is a senior member of the IEEE.

**Albert Y. Zomaya** is currently the chair professor of high performance computing and networking in the School of Information Technologies, The University of Sydney. He is a fellow of IEEE, IET, and AAAS.