JES
Journal of Engineering Sciences

Vol 11, Issue 5, MAY /2020
ISSN NO:0377-9254

Cloud of Cloud to Protect Shared Data

Dr M Suresh#1, B Yasaswini Chowdary #2, J Pravallika #3,
A Alekya #4, T Chandrika#5, Ch Yamini#6

#1 Professor & HOD, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)
#2 Student, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)
#3 Student, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)
#4 Student, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)
#5 Student, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)
#6 Student, Dept Of CSE, Qis Institute of Technology, Ongole, Prakasam (Dt)

Abstract— Cloud computing, in now days it is been playing a crucial role in terms of data storing and reducing the overall cost to entrepreneurs. But most of them worried about security; mostly they used to keep the data in single cloud. In this case if the data is lost or hacked in the sense entire data will be loose. To avoid these kinds of vulnerabilities and to achieve better security we are proposing of multicolor where the data will be stored in different databases means clouds. This paper surveys recent research related to single and multi-cloud security and addresses possible solutions. It is found that the research into the use of multicolor providers to maintain security has received less attention from the research community than has the use of single clouds. This work aims to promote the use of multi-clouds due to its ability to reduce security risks. In cloud data is been changing dynamically from user side in this case hacker may have a chance to hack the data through the network or attacking on the database.

Keywords: -Security; Distributing Data; Storing Data in Cloud multi-clouds; Single-Cloud; Data Integrity; Data Intrusion; service availability

I. INTRODUCTION

Fig.1 Cloud back up service

Cloud computing provides many benefits in terms of low cost and accessibility of data. Ensuring the security of cloud computing is a major factor in the cloud computing environment. Users often store sensitive information with cloud storage providers but these providers may be un-trusted. Working with "single cloud" providers is becoming less popular with customers due to risks of service availability failure and the possibility of data leakage. A movement towards "cloud-of-clouds", or in other words, "inter-clouds" or "multi-clouds" has emerged recently. This paper surveys recent research related cloud-of-clouds security and addresses possible solutions- one of the solutions being data distribution and replication for storage and retrieval on multiple clouds. The research into the use of cloud-of-clouds providers to maintain security has received less attention from the research community than has the use of single clouds. This work aims to promote the use of cloud-of-clouds due to its ability to reduce security risks that affect the cloud computing user. Backup file, data archival and collaboration are the popular services in cloud companies [1], in general these

services based on cloud storages like the Amazon S3, Drop box, Google Drive and Microsoft Sky Drive. These services are fashionable because of their everywhere accessibility, pay-as-you-go model, high capability, and ease of use. Such services can be generally grouped in two modules: (1) personal file synchronization services (e.g., Drop Box) - Personal file synchronization is based on back-end storage cloud model and the applications of client communicate with the local file system by monitoring interface [notify -in Linux]. (2) cloud-backed file systems (e.g., S3FS [6]). Cloud-backed file system based on two architecture models: the First model is proxy based, second model is open-source solutions [S3FS [2] and S3QL [3]]. The two models are implemented at user – level. Proxy based model the proxy component placed in network infrastructure, performing as a file server to various clients. Functionality of Core files system is implemented by proxy, to calls the cloud and stores the files. The major limitation is bottleneck and single point of failure. Open source solution model the clients directly access the cloud, exclusive of proxy interaction as a result, there is no longer a single point of failure, but it's very harder to control the file sharing between the clients when miss the suitable rendezvous point for synchronization. Cloud Backed File System: Cloud backup [4], [5] also identified by online backup, is an approach for backing up data that involves a replica of the data over a public network to an off-site system. Cloud Backed is models that provide data backed up remotely, maintained and managed. Users access the data through the network. Users normally compensate for their data storage on cloud as per-usage or monthly rate. The cloud Storage providers provide a platform as a service, is one of the infrastructure service on cloud storage to shorten storage management for enterprises and personality users.

Implementing cloud data backup is able to help boost an organization data protection without raising the workload on information technology. Online backup systems are classically built a client software application that run on a program determined by the purchase stage of service [6]. Cloud backups contain the software and hardware component to keep an organization's data, include applications Exchange and SQL Server. Online backup is used by small and medium sized businesses (SMBs) and larger enterprises to back up the data. For larger organization [7], cloud data backup as a complementary form of backup.

II. RELATED WORK

A. Distributed File System (DFS)

Distributed file system (DFS) is the file systems for cloud. DFS allows the clients to access data. Data files are separated by parts as chunks that stored on various remote systems which offer the parallel execution. Data are stored in files in the format of hierarchical tree structure. Directories are denoted by nodes. DFS facilitate any type of enterprises (such as large, medium, small) that allows storing the data and accessing the data on remotely. DFS allow two type of file system as GFS and HDFS. Both file systems are handled the batch processing. Hadoop distributed file system [8] (HDFS) designed for access the terabyte's data or peta bytes data. HDFS is master slave architecture. It consists of Name node and Data node mechanisms. Name node manages the storage of Metadata and Data Node manages the node storage. HDFS file systems the files are divided into blocks. Each block contains various data nodes and every node is replicated for availability. This is the block level replication. Name node manages the

operation of name space and map the block to data node. HDFS is characterized by method of Data re balancing. Google file system (GFS) is distributed file system that provide few faults –tolerant, high data storage and performance many clients access the data simultaneously. GFS allow the map reduce concept for access the multiple machines [9]. GFS is Single master server and multiple client architecture. The files are separated into chunks that stored in server chunk. Coordinating the resource storage and Metadata files managements is responsible for master server. In the GFS file processing done by two phases sending phase and writing phase.

B. BLUE SKY

Blue Sky is a file system of network based cloud storage. Cloud backup provide a persistent data storage, provided by Amazon s3 or windows Azure. Blue sky allows the consistency and large storage capacity and reduce use of hardware sever. Client access the cloud storage server with help of proxy running on –site. Cloud optimization is achieved by log-structured design and secure cloud log cleaner. It uses multiple protocols as NFS and CIFS for various providers. Blue sky file system is maintained by object data structures format and log structured format [8] for cloud organization. Blue sky provides version ed store data for backups in the file system. Data and Metadata is represented by blue sky object; it's presented in log structured file system [10], [11] format is data blocks, inode, inode maps and check points. For storage purpose log segments objects are aggregated. Blue sky standard file system semantics is POSIX with atomic hard links and renames. Blue sky provides a transparent proxy based structural design, with the purpose of store the data per mentally on cloud storage providers to clients in an enterprise. File data are store in data blocks. Files are separated by constant size blocks. Blocks size is 32 kb. Inodes contain the details of basic Metadata like ownership, access control timestamps, directory inode reduce the path traversals. Inode map list provide the location of inodes because inode data are not stored in permanent location. Origin of File structure snapshot is determined by checkpoint objects that locate the present inode map objects. The main use of checkpoint is managing the file system integrity. For every file structure, Blue Sky maintains a part of log for every writer to file structure. Classically there are two: the proxy managing the file system and cleaner - garbage collect data [12]. Every writer stores the log segments to separate index that provide independent updates to file system, all log contains of a number of log segments, and log segment group the objects mutually into the fixed size block. Blue Sky needed a file system cleaner to garbage collect different traditional disk-based systems contain the Flexible nature of cloud, the cleaner is not essential. Blue Sky cleaner's run on proxy. Example of blue sky cleaner provider is Amazon EC2 and S3. The log structured design allows snapshots for backup purpose. Snapshot tool record the list of checkpoints to protect the log segments from deletion.

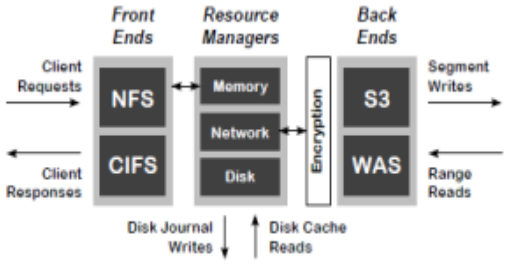
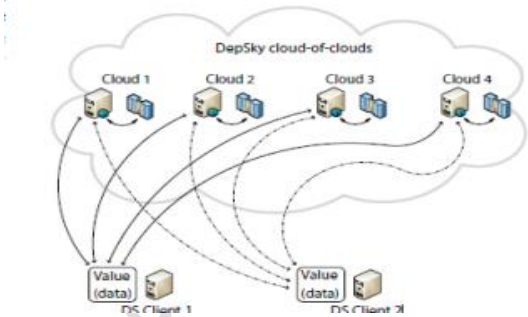


Fig.2 Back up file system - blue sky

C. DEPSKY:

Dependable and Secure Storage in a Cloud of-Clouds DEPSKY, cloud backup improve the availability, integrity and

confidentiality of information store in the cloud with help of encoding, replication and encryption of the data on varied clouds that make a cloud-of-clouds. DEPSKY is a reliable and protected storage system that gives the profit of cloud computing by using an arrangement of diverse commercial clouds to cloud-of-clouds. DEPSKY also give the virtual storage, it is accessed by users while invoking the operations. DEPSKY also provide the four limitations such as Loss and corruption of data, Loss of privacy, Vendor lock-in, Loss of availability. DEPSKY System use data and system models. It contains two main algorithms as DEPSKY – A and DEPSKY – CA and also contains the set of auxiliary protocols. Two algorithms are implemented by software library in the clients. Data model contain the three abstraction levels. First level, the conceptual data unit has unique name, version number – support the object updates, data verification – a cryptographic data hash. Second level, Conceptual unit is implemented by generic data unit, has two types of files: signed Metadata [13] file and storage file. Third level, data unit are implemented. Data unit support the operation of storage objects like creation of Metadata file, destruction of data unit, write operation and read operation. System Model uses the asynchronous distributed system; it's composed by writers, readers and cloud storage providers. Quorum protocols can provide as the backbone of storage systems. Quorum protocols contain the individual storage nodes instead of servers. Many protocols involve several steps to access the shared memory, it makes unrealistic for geographically isolated distributed systems such as DEPSKY. The DEPSKY protocols need two communication round-trips to read or write the metadata and data files. Byzantine fault-tolerant (BFT) storage is implemented by several protocols. But its require server for execute code and functions; it's not available on cloud storage. This the key difference between DEPSKY protocols and BFT protocols. DEPSKY – A is the protocol of DEPSKY, it improves the availability and integrity of storage cloud by replication using quorum techniques. DEPSKY -A include read and write algorithm.

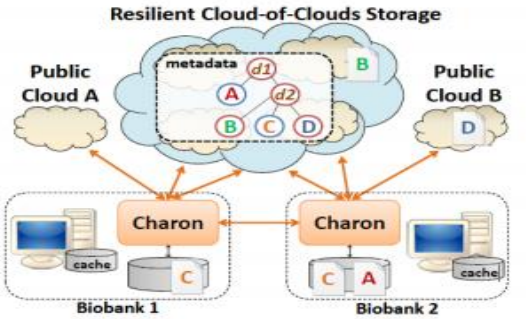


DeepSky 1

D. CHARON

CHARON is one of the cloud backed file system that able to store and share the large amount of data between various cloud providers and cloud storage system in secure, reliable manner. The two main feature of CHARON is server less design and efficient management of file system. CHAON support three types of data locations as cloud of clouds, public cloud storage and private cloud storage. Cloud of clouds provides mufti cloud availability, confidentiality. Single storage cloud is low cost compared to cloud of clouds but it requires confidence provider. Private cloud storage based on adopted method and solution, also provides the dependability level. CHARON data are separated by file data and Metadata. Metadata are stored in cloud of clouds. CHARON use data centric. Byzantine-resilient leasing algorithm which ignores the concurrency conflicts. CHARON divides the files into constant size blocks. Files are stored in various data location based on the requirements. POSIX interface is provided by CHARON that allow the user interact with any file system.

CHARON cloud storage providers are Amazon S3, Windows Azure Storage, Backspace Cloud Files, and Google Cloud Storage. CHARON consists of two design concepts: first design is writes on absorbs file and the second design is remove write – write conflicts and mechanism of ruling out optimistic. CHARON design implementation has main three challenges a: 1) Ability to deal multiple cloud storage locations, 2) Proper file system management and 3) concurrent access to the file system. CHARON use modular based approach for non fault tolerant, that build service of cloud.



III. LEASES IN THE CLOUD-OF-CLOUDS

Leases are time-based contracts used to control concurrent accesses to resources while preventing version conflicts [3]. In this section, we describe the novel Byzantine-resilient leasing algorithms we use to avoid write-write conflicts in CHARON. Besides its resilience to Byzantine failures, a defining innovation of our construction when compared to well-known lease algorithms used in practice (e.g., the ones on top of Apache Zookeeper [3]) is our data-centric design: we require no custom code deployed on the clouds. The formalization and correctness proofs of these algorithms are present in the Supplemental Material. The lease algorithm follows the system and threat model stated in (2), with a few additional assumptions. In this section, we refer to cloud services as base objects. A client may invoke, on the same base object, several parallel operations that are executed in FIFO order. Since a lease implies timing guarantees, an upper bound interval is assumed for the message transmission between clients and base objects. However, this assumption is required only for liveness since safety is always guaranteed. Our algorithm ensures that at most one correct client at a time can access the shared resource and only for a limited duration (the lease term [7]). Each resource provides three lease-related operations: lease (T) and renew (T) acquires and extends the lease for T time units, respectively, while release() ends the lease. These operations satisfy the following properties:

• Mutual Exclusion (safety): There are never two correct clients with a valid lease for the same resource.

• Obstruction-freedom (liveness): A correct client that tries to lease a resource without contention will succeed.

• Time-boundedness (liveness): A correct client that acquires a lease will hold it for at most T time units unless the lease was renewed.

Our algorithm satisfies obstruction-freedom instead of stronger properties to pursue better performance in contention-free executions[14] since write-contention is expected to be infrequent. Additionally, the time-boundedness property ensures that, if a client does not release a valid lease (e.g., due to a crash), the lease will be available again after at most T time units.

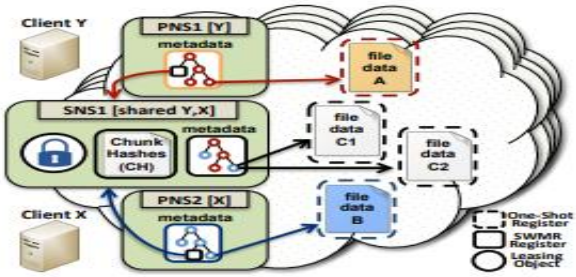
IV. IMPLEMENTATION

Metadata Organization Metadata is the set of attributes assigned to a file/directory (e.g., name, permissions).

Independently of the location of the data chunks, CHARON stores all metadata in the cloud-of-clouds using single-writer multi-reader registers to improve their accessibility and availability guarantees (see §2.3). More specifically, we redesigned and optimized the SWMR register implementation of DepSky [15] to improve the performance and concurrency as described in the remaining of this section.

A. Managing namespaces

All metadata is stored within namespace objects, which encapsulate the hierarchical structure of files and directories in a subdirectory tree. CHARON uses two types of namespaces: personal namespace (PNS) and shared namespace (SNS). A PNS stores the metadata for all non-shared objects of a client, i.e., files and directories that can only be accessed by their owner. Each client has only one associated PNS. On the other hand, a client has access to as many SNSs as the shared folders it can access. Each shared folder is associated with exactly one SNS, which is referenced in the PNSs of the clients sharing it. Although similar, personal and shared namespaces differ in the way the hashes of the most recent versions of the files' data chunks are stored. These hashes are primarily used to validate the cached file chunks. In PNSs, these hashes are serialized together with the rest of files' metadata before they are stored in the clouds. In the case of SNSs, the hashes are stored in a separate Chunk Hashes (CH) object, explained in the next section. Another difference between a PNS and a SNS is that the latter is associated with a lease to coordinate concurrent write accesses between different users which must be acquired before any update is executed on a file or directory in the namespace. Since each SNS is associated with one shared folder, a lease object is obtained to coordinate concurrent write accesses on an entire shared folder. Contrarily, concurrent readers do not require locking because the SWMR register used to store the metadata supports multiple readers even with concurrent writes. Figure 2 depicts how a set of files relates to these namespaces. Files A and B are private to their owners (Clients Y and X, respectively). File C is divided into two chunks and is shared among these clients. Since files A and B are private, their metadata is kept in their owners' PNSs. In the case of file C, the reference to the file chunks is stored in SNS1.



B. Data Management

This section describes the most important techniques CHARON uses to manage big files efficiently. Multi-level cache uses the local disk to cache the most recent files used by clients. Moreover, it also keeps a fixed small main-memory cache to improve data accesses over open files. Both of these caches implement least recently used (LRU) policies. The use of caches not only improves performance but also decreases the operational cost of the system. This happens because cloud providers charge data downloads, but usually uploads are free as an incentive to send data to their facilities [5], [5], [2], [3]. It means that the CHARON operational cost is roughly the cost of the data storage plus the traffic necessary to download new versions of files. Working with data chunks Managing large files in cloud-backed file systems brings two main challenges. First, reading (resp. writing) whole (big) files from the cloud is impractical due to the high downloading

(resp. uploading) latency [24]. Second, big files might not fit in the (memory) cache employed in cloud-backed file systems for ensuring usable performance [23], [24], [4], [7]. CHARON addresses these challenges by splitting (large) files into fixed-size chunks of 16MB, which results in blocks with a few megabytes after compression and erasure codes. This small size has been reported as having a good tradeoff between latency and throughput [15], [24], [10]. A chunk with few megabytes is relatively fast to load from disk to memory, can be transferred from/to clouds in a reasonable time, and is still small enough to be maintained in main memory. Additionally, this approach is also cost-effective because, if a cached file is updated, only the modified data chunks need to be uploaded. Figure 4 illustrates how CHARON manages and stores chunks in the clouds. After closing a file, all the new and updated chunks are inserted in a queue, from which they are consumed by a set of k threads. These threads are responsible for performing a sequence of compute-intensive operations on the chunks before uploading them to the clouds. First, we compress and encrypt the chunk. Next, we use a storage-optimal erasure code to generate 3 f + 1 distinct coded blocks of the chunk, each with 1 f+1 of its original size, in such a way that any set of f + 1 blocks can be used to reconstruct it. After that, the threads use N workers on a thread pool to send the blocks to all the clouds. Blocks already have their cloud destination assigned when they arrive in these workers (as illustrated by the patterns in this figure). The threads consider a write operation complete only when they are notified that 2 f +1 blocks were successfully uploaded. We define a timeout for workers to finish these uploads to tolerate cases in which the clouds take too long to complete operations. If some worker does not perform some operation within the stipulated timeout, the thread using this worker cancels it and launches a new one to perform the same job. In the case of a single cloud or a private [16], [17] repository is used, we do not employ erasure codes. Instead, after the chunk is encrypted and added to the thread pool, it is sent to the respective location by a single worker. In this case, if the storage location is unreachable, the data stays inaccessible. Since we parallelize [18], [19] the transmission of all data blocks and wait for the first 2 f + 1 clouds to complete, we always take advantage of the per-operation fastest clouds. This strategy aims to optimize the performance of uploading data to the clouds. It outperforms other systems' strategies in which they depend on the slowest clouds at some point [20], [22] or do not use techniques like compression or erasure coding [21]. Additionally, the configuration capability of all these architectural elements allows clients to adapt the behavior and bandwidth consumption of system's cloud-related operations with their own computing and network specificities. Moreover, employing preferred quorum techniques could adapt the system by first sending requests to clouds that best match user preferences beyond latency (e.g., cost, provider, geographical constraints). Prefetching the prominence of sequential reads in most big data workloads motivates the use of chunk prefetching. CHARON uses a thread pool for prefetching data chunks from any location as soon as a sequential read is identified. The system starts prefetching data when half of a chunk is sequentially read. If in the meanwhile the file being perfected is closed, all its prefetching tasks are canceled. Besides reading data in advance, an additional advantage of this technique is to have several TCP connections receiving data in parallel, accelerating the download of the whole file.

C. Cloud-backed Access Control

We implement a security model where the owner of the file pays for its storage and is able to define its permissions. This means that each client pays for all private data and all the shared data associated with the shared folders he created (independently on who wrote it). CHARON clients are not required to be trusted since access control is performed by the cloud providers, which enforce the permissions for each object. Moreover, the cloud-of clouds access control is satisfied even if up to f cloud providers misbehave. This happens because if an object is read from up to f faulty providers, no useful information will be obtained (recall that data is encrypted and keys are stored using secret sharing in a SWMR register). The implementation of this model requires a mapping between the file system and cloud storage abstractions. When configuring the system, users define the API credentials of the 3 f +1 clouds CHARON will use. In this way, when a COC client starts, it authenticates in each one of the providers, and after that each file or directory a user creates results in the creation of one or more objects associated with user cloud accounts. Sharing is allowed only at the granularity of directory subtrees. To give others the permission to access a directory, a user only needs to change the POSIX ACL associated with the desired directory. When this happens, COC transparently maps these permission changes in the clouds access control mechanisms using their APIs [25]. Namely, to share a directory, the system creates a SNS and gives permission to the grantee users, updating also the owner's PNS. Each COC user identifier is mapped to the corresponding cloud accounts identifiers. This mapping is kept together with the client PNS in the cloud-of-clouds. Since there is no centralized server informing clients about the arrival of other clients to the system, the discovery of new clients and shared directories has to be done by external means such as mail invitation, as in DropBox. For example, in a biobank federation, the biobanks must know the cloud account identifiers from each other.

EVALUATION We evaluate CHARON and compare it with other systems. The experiments present results of (1) the latency of the leasing algorithms, (2) several micro benchmarks of metadata and data intensive operations, and (3) a bioinformatics benchmark.

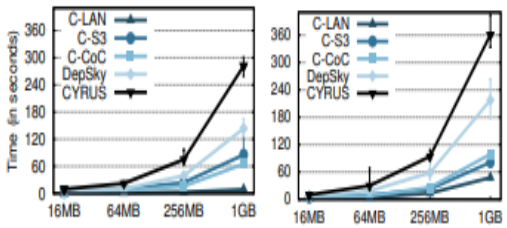
V EXPERIMENTAL ENVIRONMENT

We used four machines (Intel Xeon E5520, 32 GB RAM, 15kRPM HDD, SSD) connected through a gigabit network located in Portugal. The three storage locations for CHARON were configured as follows. The cloud-of-clouds storage uses Amazon S3 (US), Windows Azure Storage (UK), Rackspace Cloud Files (UK), and Google Cloud Storage (US). For the single cloud storage, we use only Amazon S3 (US). The private repository was either located in the client's machine disk or in a different machine in the same LAN. For the composite lease, we use additional cloud services: Azure Queue [18], Rackspace Queue [24], Amazon DynamoDB [26], and Google Data store [26]. Therefore, all cloud of-clouds configurations consider f = 1. We compare CHARON with the ext4 local file system, a Linux's NFSv4 deployed in our cluster, and other cloud-backed file systems with the code available on the web (e.g., SCFS [24] and S3QL [14]). CHARON and the other cloud-backed systems were configured to upload data to the clouds in the background. In the case of SCFS [24], the coordination service replicas were deployed in four medium VMs on Amazon EC2 (UK).

Vol 11, Issue 5, MAY /2020
ISSN NO:0377-9254

Table 2
Metadata-intensive microbenchmark results (ops/s).

Operation	ext4	NFS	S3QL	SCFS	CHARON
Create	2618	192	105	2	485
Delete	1895	2518	486	4	1258
Stat	15299	20881	5995	9	12925
MakeDir	14998	16664	4242	14	13665
DeleteDir	11998	6785	950	5	8665
ListDir	18759	17426	604	6	9894



(a) Non-cached read. (b) Write and upload.

VI CONCLUSION

CHARON is a cloud-backed file system for storing and sharing big data. Its design relies on two important principles: files metadata and data are stored in multiple clouds, without requiring trust on any of them individually, and the system is completely data centric. This design has led us to develop a novel Byzantine resilient leasing protocol to avoid write-write conflicts without any custom server. Our results show that this design is feasible and can be employed in real-world institutions that need to store and share large critical datasets in a controlled way.

VII. REFERENCES

[1] Abraham, G. Chockler, I. Keidar and D. Malkhi, "Byzantine disk paxos: optimal resilience with Byzantine shared memory", Distributed Computing, 18(5), 2006, pp. 387-408.

[2] H. Abu-Libdeh, L. Princehouse and H. Weatherspoon, "RACS: a case for cloud storage diversity", SoCC'10:Proc. 1st ACM symposium on Cloud computing, 2010, pp. 229-240.

[3] D. Agrawal, A. El Abbadi, F. Emekci and A. Metwally, "Database Management as a Service: Challenges and Opportunities", ICDE'09:Proc.25thIntl. Conf. on Data Engineering, 2009, pp. 1709-1716.

[4] M.A. AlZain and E. Pardede, "Using Multi Shares for Ensuring Privacy in Database-as-a-Service", 44th Hawaii Intl. Conf. on System Sciences (HICSS), 2011, pp. 1-9.

[5] Amazon, Amazon Web Services. Web services lic ensing agreement, October3,2006.

[6] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores", Proc. 14th ACM

[7] Cloud Harmony, “Service Status,” https://cloudharmony.com/status-of-storage-group-by-regions, 2019.

[8] Cloud Security Alliance, “Top Threats,” https://cloudsecurityalliance.org/group/top-threats/, 2016.

[9] M. A. C. Dekker, “Critical Cloud Computing: A CIIP perspective on cloud computing services (v1.0),” European Network and Information Security Agency (ENISA), Tech. Rep., 2012.

[10] H. S. Gunawi et al., “Why does the cloud stop computing?: Lessons from hundreds of service outages,” in Proc. of the SoCC, 2016.

[11] European Commission, "Data protection," https://ec.europa.eu/info/law/law-topic/data-rotection en, 2018.

[12] G. Gaskell and M. W. Bauer, Genomics and Society: Legal, Ethical and Social Dimensions. Routledge, 2013.

[13] Bessani et al., "BiobankCloud: a platform for the secure storage, sharing, and processing of large biomedical data sets," in DMAH, 2015.

[14] H. Gottweis et al., "Biobanks for Europe: A challenge for governance," European Commission, Directorate-General for Research and Innovation, Tech. Rep., 2012.

[15] P. E. Verissimo and A. Bessani, "E-biobanking: What have you done to my cell samples?" IEEE Security Privacy, vol. 11, no. 6, pp. 62–65, 2013.

[16] P. R. Burton et al., "Size matters: just how big is big? Quantifying realistic sample size requirements for human genome epidemiology," Int J Epidemiol, vol. 38, no. 1, pp. 263–273, 2009.

[17] D. Haussler et al., "A million cancer genome warehouse," University of Berkley, Dept. of Electrical Engineering and Computer Science, Tech. Rep., 2012.

[18] R. W. G. Watson, E. W. Kay, and D. Smith, "Integrating biobanks: addressing the practical and ethical issues to deliver a valuable tool for cancer research," Nature Reviews Cancer, vol. 10, no. 9, pp. 646–651, 2010.

[19] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A case for cloud storage diversity." SoCC, pp. 229–240, 2010.

[20] C. Basescu et al., "Robust data sharing with key-value stores," in Proc. of the DSN, 2012.

[21] Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa, "DepSky: Dependable and secure storage in cloud-of-clouds," ACM Trans. Storage, vol. 9, no. 4, pp. 12:1–12:33, 2013.

[22] T. Oliveira, R. Mendes, and A. Bessani, "Exploring key-value stores in multi-writer Byzantine-resilient register emulations," in Proc. of the OPODIS, 2016. Amazon, "Amazon S3," http://aws.amazon.com/s3/, 2019.

[23]Microsoft, "Microsoft Azure Queue," documentation/articles/storage-dotnet-how-to-use-queues/, 2019.

[24] B. Martens, M. Walterbusch, and F. Teuteberg, "Costing of cloud computing services: A total cost of ownership approach," in Proc. of the HICSS, 2012.

[25] J. Y. Chung, C. Joe-Wong, S. Ha, J. W.-K. Hong, and M. Chiang, "CYRUS: Towards client-defined cloud storage," in Proc. of the EuroSys, 2015.

[26] S. Han et al., "MetaSync: File synchronization across multiple untrusted storage services," in Proc. of the USENIX ATC, 2015.

Vol 11, Issue 5, MAY /2020
ISSN NO:0377-9254

Authors Profile

Dr M Suresh, M.Tech., Ph.D working as Professor & HOD in the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**

B Yasaswini Chowdary pursuing B Tech in the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**

J Pravallika pursuing B Tech in the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**

A Alekhya pursuing B Tech the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**

T Chandrika pursuing B Tech the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**

Ch Yamini pursuing B Tech the Department of Computer Science & Engineering in **Qis Institute of Technology, Ongole, Prakasam (Dt)**