Survey Paper on Security & Privacy Issues in Cloud Storage Systems

Anup Mathew
The Institute for Computing, Information and Cognitive Systems (ICICS),
University of British Columbia
Vancouver, BC V6T 1Z4
Canada
anpmat@interchange.ubc.ca

Abstract
We are increasingly relying on a number of online file storage systems to back up our data or use it as a collaborative tool in real time. All these services bring with it a fair share of security and privacy vulnerabilities for all the conveniences provided by them. In this survey paper, I seek to describe the various issues related to data security, privacy and availability with storing data on third party service providers, more commonly termed as cloud service. There is a lot of research being done to point out issues with these service providers and cloud security in general. In this paper we look at the various current researches being done to solve these issues, the current trends in securing, ensuring privacy and availability of these data on cloud storage services.

Categories and Subject Descriptors
Cloud Storage, Provable data Possession, Privacy in Cloud Storage, Cloud data availability

General Terms
Security, Reliability

Keywords
Cloud Storage, Security, Privacy, Data Availability, Provable data possession

1. Introduction
In this day and age it is all but natural for most of us to have an account in one of the many online file storage applications which we use to easily retrieve the files we uploaded, practically almost anywhere in the world on any of the many devices that we may own. We are reaching a point where we are, as I stated in the abstract, increasingly reliant on these services to be productive working individually and when working in a collaborative environment. There is a need to share artifacts and information in real time but as with many other technologies these benefits come with a fair amount of assumptions about security and privacy, which should be properly understood before we completely surrender one's data to these service providers.

As with any storage system, there are certain security properties that are desirable in a cloud storage system: confidentiality, integrity, write-serializability and read freshness. These properties ensure that user's data is always secure and cannot be modified by unauthorized users and the data is always at the latest versions when being retrieved by the user. [3]

The data not only need to be protected during transmission but also when the data is stored in the service provider's storage or hardware and in order to do this different service providers provide various levels of security and privacy for the data stored based on the resources available to them like bandwidth, cost of operations, data availability claims and business priorities. At one extreme could be service providers which provide absolute data security, top notch encryption and ensuring the user's data being accessible only by the user and no one else which might lead to issues of provable data ownership which can be exploited by user to use the service as an online slack space or if the user accidently forgets his access credential there might not be a way to retrieve the data causing data availability issues. On the flipside there might be another group of service providers who provide relatively acceptable measures of security and privacy on the data being stored but provide excellent guarantees on data availability with versioning and fast data syncing albeit exposing certain vulnerabilities in some extreme threat models.

In many of these services there has been countless instances of issues, an example of this was the recent Dropbox issue where user's data were exposed due an error in a code update [7] and in other cases the design of the online storage system itself could have allowed malicious agents to glean considerable amount of data from these services before the attacks were discovered [1] and still in other cases some of the service providers transfer the onus of securing the data on the cloud completely to the user, which might cause issues with data availability if the user forgets his or her secret token to access this data [8].

There are similar issues faced by enterprise users too, but the focus specifically being on how to best combine features of security, privacy and availability of data. Most of the enterprise users shy away from cloud storage because of lack of credible security features and clear specifications of failure scenarios and how recovery can be done. There is a

need for these users to have access to their data without being reliant on the service provider's design and implementation. Another feature that is desirable is data recovery with good versioning systems so that previous data is always recoverable in most of the circumstances conceivable or otherwise. If the enterprise users actually deploy their applications on top these service provider's data storage solutions there is an implicit need for the service to have good latency, overhead and staleness. [2] These systems expect the data to be used to detect if there was a violation of integrity, write serializability and freshness along with features that can be used to prove that any of these violation can attributed to the third party.

All these issues only reaffirms the fact that a lot of effort and research both on part of the academia and the industry need to be put into providing a secure as well as reliable cloud storage system. In this survey paper we look at some of the issues in providing secure cloud storage services. When specifying the term secure it means features such as confidentiality, availability of data, providing integrity of the data and ensuring trust between the provider and the user. We would be discussing solutions to these issues and how feasible it is in a cloud storage industry setting by gaining a clear understanding of present cloud storage provider's technology.

2. Discussion

We will start the discussion in section 2.1 where we will discuss the basic terminologies and aspects of cloud computing infrastructure, so as to better understand the issues then we will consider the various issues that could compromise and complicate provision of security and privacy of data in cloud storage in section 2.2; we would then take a look at various research solutions and see how well they solve some of the problems in section 2.3.

2.1 Security and Privacy in Cloud

One of the core themes of cloud computing and cloud storage in general is that service should be independent of the location. [39] Some of these aspects affect the way the cloud service provider creates his service and might lead to security and privacy issues for the consumer of the services. Some of the characteristics of the infrastructure are detailed as follows.

Location Independent Services: The very characteristics of the cloud computing services is the ability to provide services to their clients irrespective of the location of the provider, the physical hardware below could be moved anywhere but the services should still be available. This feature also applies to the consumers of the services, with the advent of mobile computing platform the consumption of the services cannot be restricted to a particular location but may be requested from any location as per the choices of the customer.

Communications: Due to the very nature of the cloud computing infrastructure communications is a major component in every design. These communication lines could exist from few seconds to hours based on the services being consumed. So the security of this communication lines should be persistent as long as the connection between the provider and consumer exists at minimum and cover some buffer period too.

Infrastructure: The infrastructure that is used for these services should be secured appropriately to avoid any potential security threats and should cover the life time of component. This lifetime can be estimated to be about 10 years.

Storage Security: The data that is stored on the cloud services often would last longer than the security that should be ensured of the components which are used to store or compute these data. This would entail the storage services should be robust enough to achieve component and hardware changes easily and transparently. This applies to the algorithms and encryptions schemes that are used to secure this data; they could become obsolete and might become easy targets to brute force attacks as the processing powers of the various devices keep increasing.

Backup Storage: In this aspect the security should outlast general storage security and the life span could be assumed to be greater than thirty years, and as with normal storage services the technologies should be resistant to component and hardware changes as well as the algorithms used to store the data.

Security issues could be classified into two parts based on the points at which these threats are possible.

Access Security: Communications to the cloud service provider is a potential point at which threats to the service could be exposed. A lot of research has gone into securing communication channels and have proved quite resilient to the threats that we come across. Though with the advent of mobile computing systems a potential threat to the security and possibly privacy of the users would be location security as this would entail the presence of communications to identify the location.

Service Security: In these scenarios most of the security threats are possible at the point of service provision and this could include the actual device security at the cloud provider and the storage security used by the provider. Though due to the business nature of the service providers they would be able to provide robust security with the use of state of the art IDS, firewalls and malware protection. Moreover the use of virtualization technology further helps the providers in securing each of the individual users from each other.

As for the privacy aspects of cloud computing it is quite complicated to general quantify

privacy and use it as a means to come to decision points. Often the cause of this is due to the asymmetrical nature of privacy to the users and cloud service providers when compared to other service providers. Data deemed as private to users might be actually very valuable to the cloud service provider and acts as means for providing extra revenue options and this in turn may lead to service becoming cheaper and possibly better. Another aspect to the whole privacy conundrum is the fact that privacy is like the Barn's Door, once the sensitive private information has been leaked it would be very tough for the user to control the effects of those revelations.

There are various stake holders in a cloud service environment and they are explained below:

Individual Users: This is a huge number of individual consumers who use the services provided by the service providers and are one of the primary targets for the features. These users may not have privileges to influence the features of the cloud service directly.

Aggregate Users: These are users in a group such as organizations or corporates and are usually managed and controlled by a common management authority inside the respective organizations. This authority have fair amount of control over what services are being provided by the cloud service provider and negotiate to receive service tailored to their need specifically.

Cloud Service Providers: These are service providers which would provide on demand services such as computing, storage or other related services.

Figure 11 provides a rough overview of the cloud service provider ecosystem [39] and describe how the service could span jurisdictions and the same consumer could be mobile and could be in different jurisdictions.

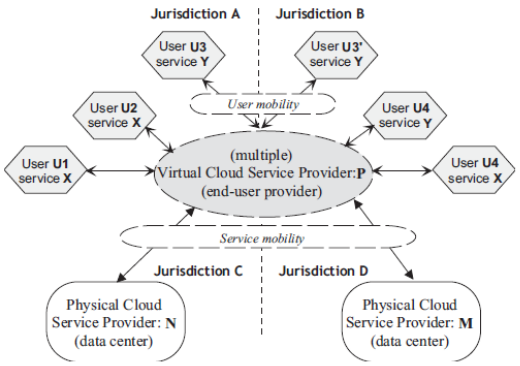


Fig.: 11 Cloud Architecture

Although since cloud service tends to be location independent it could lead to many legal and jurisdiction based issues since the service may not be limited to any one boundary and due to this privacy implication on the data could vary among the various jurisdictions. Another term that is often used in cloud computing terminology is private cloud and public cloud. Private cloud service providers do not expose the service to external public and provide in house service to organization and corporates, whereas public cloud service provider provides the services as commodity to everyone in the market.

2.2 Issues

There are various types of issues that a cloud storage user both at enterprise level and as an individual consumer might face during the use of the service. Most of the issues are with integrity of the data, ensuring that the data is confidential and available when it is needed. Let us look at these facts in a more detailed manner. This is not an exhaustive list but certainly covers some of the more urgent and significant issues.

2.1.1 Trusting data stored in the cloud

Data when stored in the cloud needs to be not only be confidential but also should be correct every time it retrieved after uploaded or after a modification, there should not be a loss of integrity of the data. This is a valid scenario when third party storage services are compromised by the malicious agents, the data that is being provided by the corrupted service might not be correct or fresh. This can be sometimes very hard to detect and can sometimes lead to considerable information leakage before being discovered hence certain amount of onus lies on the service user to trust the provider that what he provides is correct inside the boundary of integrity check guidelines that have been agreed upon between the service provider and the user, but which might be not be correct when the service provider's infrastructure has been compromised or encountered an error. To an extent this problem is due to the fact that service providers may sacrifice trust for providing liveness and high data availability. [2]

2.1.2 Lack of provable security in Cloud Service provider agreements

As we move towards a more pervasive use of the cloud storage service it will become more of a commodity business, security would be needed and be necessary to differentiate service providers and systems. This is not the case right now in the industry since most of the cloud service providers today provide service level agreements with emphasis on high data availability with little guarantee on the security of the data. [3] Due to internal errors or sometimes malicious changes to their system the data might be exposed or provided to the users of the system with the integrity being compromised. This trend does not help the customers using the service to prove that their data has been compromised if and when this happens.

2.1.3 Data history

One of the significant features we enjoy with local data storage is the presence of metadata features

which allow us to view the history of a data object. This allows the systems to provide data integrity checks and rollback capabilities when a corruption or compromise is detected in the system. These features are almost non prevalent in the present cloud system and if present there are substantial security vulnerabilities associated with it because of the scale of the service. This feature which has become de facto for ordinary storage system on local systems and provided by most of the data storage systems needs to be implemented in the cloud service context effectively considering the scale of the system.

2.1.4 Provable Data Possession

This issue is loosely related to one of the other issues we looked into on how to trust the data stored on the service provider. When a data is retrieved from the service provider on performing an integrity check, it would be very hard to determine how the data was stored in the service providers system. This is to ensure that the data is not leaked to a third party to whom the service provider is outsourcing the data, when the agreement for service is being agreed upon by the service provider and customer. The present service providers provide hardly any sort of security on where and how the data is being stored and how secure the systems are vis-à-vis the claims by the service provider.

2.1.5 Use of Cloud Storage service as an online slack space

The cloud service providers have various business objectives when providing the service based upon these the product might be designed. If one of these priorities is to provide quick syncing capabilities, then some sort of system design must be provided so as to not sync data that is already present in the cloud storage even if it belongs to a different user. This leads to a situation wherein the link between a chunk of data and user might not be direct, this would allow user to modify the system to use storage space anonymously without provable data ownership. This could lead to various legal issues if this service is used to provide illegal services.

2.3 Research

2.2.1 Untrusted cloud storage

Cloud storage in all its spirit has made it easy, scalable and always available data storage with feasible cost reductions though with all the said features the client needs to completely trust the provider and during states of buggy software, hardware failures and malicious attacks the service might encounter inconsistent service and it is only in the best interest of the client to be safe from these issues. There have been many researches in recent times on ensuring this property and a very feasible and valid solution to this problem is called Depot [2]. The technique aims to achieve safety of data by eliminating trust from the equation and assuming that the client's systems are correct always and minimizing the trust on service provider's infrastructure and as a result improving liveness and availability of the data.

Threat Model:

The nodes are assumed to trustable at the clients side and all the nodes including the servers are assumed to be suitably cryptographically hardened. The nodes might fail due to any reason and it could be hardware failures, data corruption or malicious attacks by external agencies. Two properly function nodes would eventually be able to communicate with each other to exchange updates and information, hence the assumption is made that the compromised nodes would not be able to hinder this communication for a persistent duration. A node might crash and recover to the correct state up until the time when it comes online and this capability is always assumed to be true. The nonfunctioning compromised nodes are eventually identified and are assumed to be removed from the system or their issues are corrected eventually.

Some of the aims that the system achieves are as follows: The updates that are sent by the clients to the servers are all signed and provide previous system updates and system state during the update. This way if a client or server notices forks the system provides options to merge the forks. Enforcing FJC or fork join consistency allows for the system to be highly available since FJC is slightly weaker than causal consistency. The following figure shows the architecture of Depot adapted from the author's paper [2].

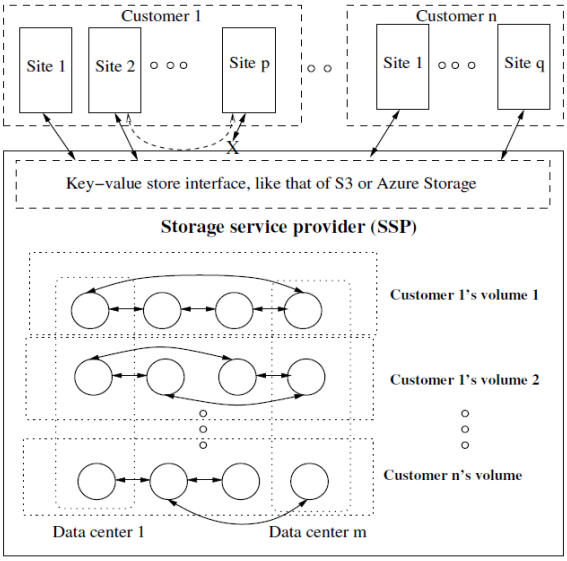


Fig: 1 Depot's Architecture

The clients store the data as objects by providing a key to each of the object that needs to be stored and sending the pair to the server nodes. The storage system is grouped together and each group is

responsible for one of or more volumes and each volume might contain a single customer's objects or multiple customers' data based on the key value's range. The key partitioning decides the number of volumes and which volumes get which range of keys. The key fact about this system is that in the eventuality that none of the servers are available the clients could contact each other directly for the objects for the given key value. Some of the issues that are answered by this technique are:

Consistency: The system allows maximum consistency by adjusting the order, delay or updates to provide correct reads. The updates by the clients are sent in the following form [2]:

$$\text{dVV}, \{\text{key}, \text{H(value)}, \text{logicalClock@nodeID}, \text{H(history)}\}_{\sigma_{\text{nodeID}}}$$

Each update is associated with a logical clock with the corresponding node id assigned to the node and every time a write is done the clock is incremented. When another node receives this update it increments its own clock to exceed the value specified in the update's logical clock. The other fields in the update are the hash value of the object and hash value of the history in the node sending the update and the signature of the client sending the update to avoid faulty updates. Each stores two data structures containing the updates it has done and the checkpoints which are corresponding states of the node. For a properly functioning node to correctly carry out the received update these conditions must be satisfied. The update must be properly signed, the update must be newer than previous updates the node sent, the nodes should have the update's dVV, and the hash value of the history in the update should match the hash value of the history the node calculated thus ensuring the updates are ordered according to the correct order. The update's time stamp should be a constant times the value of the updating nodes current clock time. Using these techniques Depot tries to achieve consistency in data during updates.

Data availability: The system ensures the availability of data by returning the read value if there is any reachable and correct node. Additionally it also ensures that if a version of the object is available in any of the correct nodes before the object is removed by the garbage collector then the value of the object is returned by the read ensuring durability of the read operation.

Data integrity: The data objects should never be updated by unauthorized clients and in order to achieve this limitation the system ensures that only correct and authorized client are able to perform the updates. Each of the volume is configured at start to be associated with a particular range of keys to particular public keys of nodes, so effectively only a subset of the client's object collection can be updated by any particular node.

Data recovery: Like any data storage system in the eventuality of data corruption or data loss there should be provisions to recover the data from the failure and this system with its use of FJC features provides reliable provisions to roll back or recover data. The approach the system takes is basic ladder back up technique i.e. all the versions of an object are kept for a day, single version are kept for each day for a week; a single version per week is kept for a month and subsequently a single version per month for a year. The servers and clients keep each update that is received and created in case of clients. Every day one client is selected for carrying the cleanup of the backups and non-laddered versions of the backups are removed by unanimous resolution between the clients and the corresponding client would create a candidate discard list [2] which would indicate all the old checkpoints that need to be removed or discarded.

Evicting faulty nodes: The system uses the update's signatures to verify misbehaving clients and eventually evict them from the system.

As discussed this system provides a reliable way to ensure that data provided by the system is always available to the client even during the eventuality of the systems having faulty nodes and data becoming corrupted in certain subsets of the nodes. The system provides reliable guarantees to the data being provided by the service provider and gives robust back up capabilities.

The advantages of this system over other similar approaches are that replication of data over different machines would tolerate failures only up to a fraction of the total machines fail and as for data reliability using forks the issues of liveness of data becomes apparent when a server with issues permanently forks certain correct clients permanently.

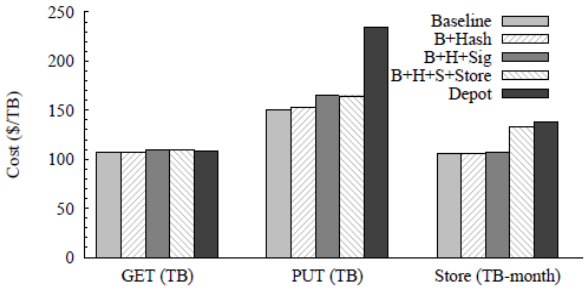


Fig 2: Dollar Cost per TB

The cost wise advantage of implementing the system also provides a good reason for feasibility of this technique. The costs to transfer one GB of data to the central depot and local nodes are depicted above. [2] This provides considerable emphasis on fault tolerance level without sacrificing availability, and this is their main advantage compared to other system proposed by others as in. [7, 8, 9, 10] As expected

in the other range of system those which provide very high fault tolerance capability but with limited data availability and data liveness guarantees causing the system to be impractical. [11, 12, 13] Still in other systems which provide fork based techniques the system have extremely high fault tolerance levels but very limited availability causing the system to be again impractical. [14, 15]

2.2.2 Reliable Cloud Storage and Data History

Cloud storage is being used as a means to store backups of the local systems and other user data or application data, but a very important property that we have come to be associated with the desktop storage system which provides a suitable provision for data security is missing in the cloud storage systems, data provenance. A very good example of this situation would be the instance of a cloud storage system containing many nodes storing digital astronomical data from telescopes and other space imaging systems.[16] Due to usage of the data being used in bursts and if a particular node's data is modified without the knowledge of other users the data generated using these modified objects would be inconsistent and if no sufficient meta data stating the provenance of the data is present the discovery of what exactly went wrong would be a tedious task. The same principle applies to data in the cloud, which is constantly at the risk of inconsistencies and corruptions and the client receiving the data need some sort of provenance of the data to determine if an issue is discovered due to a malicious attack on the data on cloud. There are various provenance solutions that has been proposed though none have been suitably adapted to function properly with the cloud which has a primary property of high availability and scalability.

Provenance of data objects in a cloud storage environment is extremely important because data on the cloud in most scenarios would be shared and often widely and this makes it important that data consumers should have the ability to know how the data was updated and how trust worthy the data is, especially in scenarios where data is corrupted or is not what is actually expected the system could detect malicious attempts at corrupting the data hence enhancing the security of these systems.

A simple way to store provenance of a data object like a file would be to create a directed acyclic graph (DAG), each node in the graph representing a file. The graph is by definition non cyclic because existence of the cycle would indicate that an object is the ancestor to itself.

A good and effective solution to this situation would be the technique proposed by Muniswamy-Reddy et al. [4] Their solution involves usage of PASS[17] system to collect provenance data during data storage operation to the cloud and updating these data in the cloud separately or as part of the data during regular intervals. The system records any system call to the storage system in the cloud by creating a wrapper to the cloud storage API calls. For example during a read request the provenance system creates an edge in the DAG graph node representing the file. PASS would record various attributes like the file name, the process name that created it, file id etc. In order for the provenance of data to be suitable for cloud storage some of these properties need to be adhered. [4]

Data-Independent Persistence: This property is used to ensure that the cloud store the data objects provenance even if the data is removed from the system.

Provenance Data Coupling: The provenance of the data object should be able to completely and accurately describe the data object and should be tightly coupled.

Multi-Object Causal ordering: This ensures the causal relationship between objects. A fair way to explain this scenario would be that if an object A is created due to operations on an input to B, then the provenance label of object A should be a superset of the object B.

The following diagram shows the architecture of the system proposed by the authors of the technique. [4]

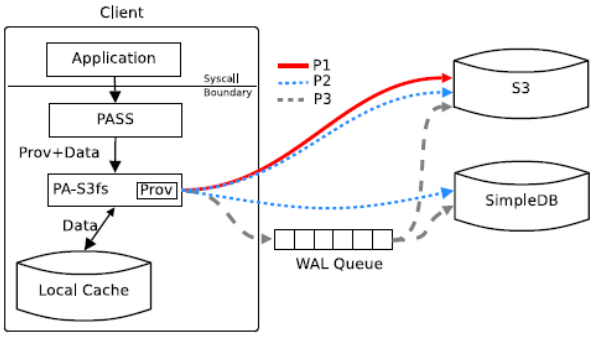


Fig 3: PASS Provenance System for Cloud Storage

The system shown above consists of the PASS provenance system on the local system which has a wrapper over the cloud storage API and is termed as PA-S3fs since it's built on top of Amazon S3 file storage service API. Whenever PASS system reads or writes data it communicates with the PA-S3fs module which caches the data and provenance data in the local storage cache and at regular intervals or when file operation events such as file close occurs the data and provenance data is sent to the cloud using once of the protocols P1, P2 and P3. These protocols are described below: [4]

P1: Standalone Cloud Store: In this protocol both the files and provenance data is stored as two separate S3 objects. This protocol has the limitation that it would delete the provenance data along with the data when it is removed and there is limit to the size of the

provenance data that can associated with the data object, both of this could be overcome by storing the provenance data in the data object itself by storing it in the first part of the object of fixed byte length.

P2: Cloud Store with Cloud Database: This protocol involves the usage of the data object in the cloud as a S3 object and the corresponding data provenance values in the SimpleDB as a single row for each version of the object, each of the versions of the object is identified by a unique user id assigned to the object. This protocol is an improvement over P1 though it doesn't satisfy the data-coupling property that is usual desired.

P3: Cloud Storage with Cloud Database and Messaging Service: In this protocol uses the cloud messaging service along with transactions to enable data coupling. The client has a log storing the read and write operation and a separate process uploads this data to the cloud when the transaction is completed.

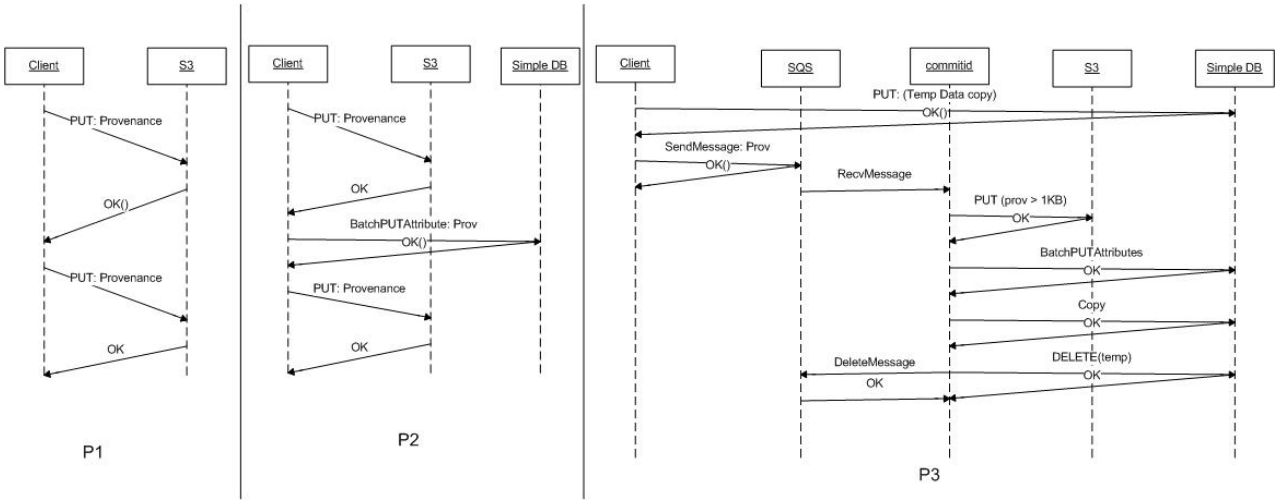


Fig: 4 Provenance Communication protocol

Fig 4[4] describes the three different protocols in a sequence diagram. The protocol discussed in the technique has been evaluated to be quite performance friendly and the workload overhead can be lesser than 10% in the benchmarks the authors performed. This is a valid solution towards providing provenance in cloud but the ultimate goal should be to provide data provenance built in to the cloud storage systems itself, so that data security breaches can be identified with enough Meta data.

The system described above pertains to ensuring that the data is available and correct even in situations when the service provider is least trusted. Recently though another facet to this problem has been researched primarily relating to how to ensure that the data that is sent to the service provider is receiving maximum security, availability and liveness. The term associated with these features are Provable Data Possession (PDP) or Proof of Data Retrivability (POR). [6] The service provider may or may not be malicious but due to their practices or due to malicious activities by others they might lose or corrupt the data. These problems have been researched in the papers on PDP [18] and POR [19].The main advantages of these schemes over traditional schemes of sending updates as encrypted data and carrying operations over encrypted data is that it require considerable CPU power and memory to perform these operation and in cases of devices with smaller resources these operations becomes infeasible and often extremely costly like in case of mobile phone or a tablet. In most of the cases a very quick and general solution to overcome this problem would be use data replication to overcome issues but this solution would be infeasible when data sizes exceed petabytes and are unnecessary overheads and become unsustainable quickly.

The whole issue of PDP and POR becomes more important when personal data is outsourced to cloud service providers like in the case of photo storage service provided by Dropbox [20] from mobile phone. The importance of data is high enough even though the user is exploiting a potentially free service and since the chances of data being accessed from the mobile phone is higher in this case the PDP scheme should be able to provide data provenance using clear text format and efficiently use computation resources and network bandwidth.

The solution proposed by Ateniese et al is described to have the following goals, Efficiency and Security - this is provided by the use of encryption schemes on the verification data and not the bulk data itself hence providing a more efficient use of the client's resources. Dynamic Data Support – this provides the client with support for data verification and additional security even if the data on the server is modified, deleted or more data is later added to the existing data.

The general technique involved in this system is that symmetric key cryptography is used to

verify the data present in the server. When the client uploads the data to the server, it computes a list of token on random data blocks in the data being uploaded. After this upload operation is complete the client may randomly request the server for these tokens over the specified data blocks, the server in turn computes these tokens using the same algorithm that the client used and then responds to the request with the tokens it calculated. The proof will hold if the data integrity tokens returned by the server is same as the tokens that were pre computed by the client before it sent the data to the server. Additionally these pre computed tokens may be encrypted and sent to the server when storing the data and this way the client's storage overhead would be constant.

Both these systems described above seek to solve the issue of data provenance and reliable service by the cloud storage service providers which can act to be suitably used by the service consumer to verify the data as well as detect any mal practice or malicious modification of his data during the period that the data is present in the service provider.

2.2.3 Provable Security in Cloud Storage Systems

As with any systems today along with providing security of the data stored there should also be a technique to show as well as prove that data is being stored with maximum available security measures. This issue is particularly important in case of enterprise data where confidentiality of the data is important and a valid display and proof the security being taken by the service to ensure the service agreement is very necessary. Often service level agreements used to ignore these aspects since there were no concrete systems or techniques to allow a quantification or measurement of this features. A lot of research has been happening in this field. [22, 23, 24, 25, 26] Most of these systems are not effective in this scenario because they were not designed for a cloud storage system and were more suitable for a personal storage system locally deployed. Along with these problem these system were extremely capable at detecting data corruption or server misbehavior but were not able to provide suitable proofs of this corruptions or misbehaviors. Another problem with those solutions was their apparent limitations toward scalability and with cloud service provider it is a very important aspect of the service they provided. Hence a very capable solution to overcome all these problems was suggested by Raluca et al. [3]

The system uses a standard API interface for put and retrieving data based on a block id assigned to each of the content block. The data owner need not be online all the time for verification of the various properties of the data. These properties of the data stored in the cloud is used by the system to show and subsequently prove data corruption, the properties are Confidentiality which ensures that unauthorized users do not access the data, Integrity which emphasizes that the data returned during each read by the client is the exact same data uploaded by itself or some other authorized client. Freshness verifies that the data returned is the latest most up to date data that has been uploaded by a client, owner or other authorized peers. Write serializability ensures that the authorized user does an update to the data only after receiving the latest updates from server. A non-adherence to the properties Integrity, Write Serializability and Freshness would indicate that there has been a security compromise. [3]

The core mechanism in which this system works is by providing and exchanging attestations among users, owners and cloud service providers to verify the violation of any of the three mentioned properties above. To prevent un-authorized reads the data on the server is encrypted with a proven block or stream cipher algorithm like AES. Each client that has access to the data will have the decryption key to access the data. Write access control is achieved by using a key that is public as a means to verify the data and a private key that is used to sign the updates that are sent to the service provider. Whenever a update needs to be uploaded to the data block the legitimate user will sign with the private key the data hash that will subsequently be uploaded and every time a update is received by other legitimate clients they can verify that the update is authorized by checking the hash value of the data by unencrypting it using the public verification key.

In this entire scheme the key distribution would turn out to be the bottle neck to the whole scheme and in order to achieve maximum efficiency the cloud service provider is used to provide the key distribution. A technique called Broadcast encryption [27, 28] is used to achieve this scheme. Using this scheme the data owner writes a data block containing a family key block, which can be only be modified by the data owner and each family have a single access control list. The data owner encrypts the read access key in this block so that only users having access to this list have access to the data in the family.

As described earlier the main principle behind allowing users to prove any cloud misbehavior is through attestations. These attestations on a high level is like certificates that certify that the client authorized over writing a data block and the cloud service provider is doing it on the user's behalf and vice versa in case of data being provided by the cloud to the user. The data structures of these attestations are shown below in the diagram in figure 5. With all these features confidentiality is achieved by ensuring clients put their data in an encrypted form. Integrity in this system is verified by the fact that every time an update is provided by the client the client must provide a signed hash and when receiving an update other clients will verify the

signed hash using the public verification key obtained from the key block.

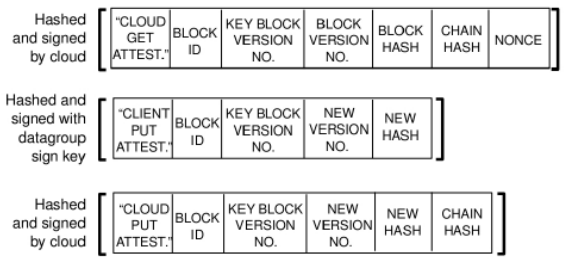


Fig: 5 Attestation Data Structure [3]

If there is an integrity violation then the integrity signature on a block would match the signature generated using the block data. As for checks of write serializability and freshness for any violation or deviation the owner of data can perform audits on the block of data periodically during every end of epochs. Each data block is assigned a probability for checks to be performed and in cases where certain data blocks are very sensitive the probability can be increased to one to have it checked every time the audit process take place after the epoch time.

When a data block needs to be attested the data owner would transfer the attestations it received from the cloud after sorting these attestations as per the change order and version number. The cloud in turn would check for attestations from the various clients and if a malicious client is present they would not send some of the attestations thus proving the fact that they are malicious and when the owner asks the attestations that are missing the honest cloud could provide it to them to detect this breach. The potential overhead in this solution is mainly due to the network latency caused due to extra set of communications that need to happen during GET and PUT requests and as such the computation overhead is inside reasonable limit. The read and write throughputs below would give an indication of the efficiency of the solution.

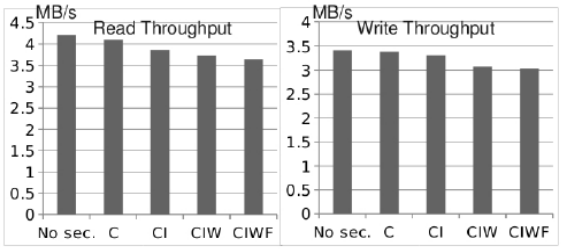


Fig: 6 Throughput of GET and PUT requests

The solution has a weakness as in a vulnerability to DOS attacks because of all the operations being done by the cloud for a single request by client. In order to avoid this potential attack the authors suggest that every time the client does a get request the client should acknowledge by providing the get attestation singed with the read access key thus avoid unauthorized clients from performing a get request on data without the valid read access key and also making get request slightly more compute intensive.

2.2.4 Error/Malicious Node Localization

As we have discussed in earlier sections that data integrity and its violation due to malicious intents can be detected at various costs as described by the researches mentioned in [29, 30, 31, 32, and 33] in a single server scenario which can be substantially extended to the cloud computing scenario and various distributed redundancy system described in [34, 35, and 36] to ensure data integrity. All these technologies do not treat the security threats from malicious agents and Wang et al [5] have proposed a technique to overcome the limitations and provide more secure storage systems for the cloud computing scenario.

The possible security threats that are possible in this scenarios could be a possibly malicious and self-interested cloud service provider due to monetary reasons might remove the data less frequently used to secondary storage devices or would try to hide a data corruption or loss incidents due to internal issues. Another possibility is well financed organization or individuals who are able to perpetuate a malicious break down of the service providers systems at various time periods and try to modify or remove the user's data from the providers systems randomly.

The idea proposed by them is a step forward in securing data as it provides a distributed verification of erasure-coded data [37] [38] and during storage correction steps the system would be able to identify the error prone server thus providing a good error localization scheme. Their extensive analysis of the system has shown that the scheme is quite effective against malicious data modification, server colluding attacks and Byzantine failures.

The basic architecture of the solution proposed is depicted below as adapted from the paper published. [5]

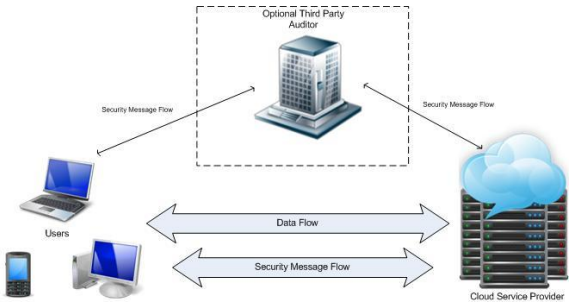


Fig: 6 Architecture

The solution involves three sets of entities which are the users, who store data in the cloud storage service providers and use some of the cloud computation services provided by them. The users could be either

enterprise users or individual consumers. The provider of the service is termed as the Cloud Service Provider and provide substantial computing resources and capable of managing and deploy large distributed architecture. An important albeit a optional part of the system is the independent third party auditors who have the capability to verify and detect risk of cloud storage service providers on request by the consumers of these services. In most of the case this third party authorities are needed if the users themselves do not have the resources or time to carry out the audit of the cloud service providers.

The general idea to check for the data correctness is initially before the data is uploaded on to the cloud and distributed redundantly the user generates short verification tokens, each token being generated from a random set of data blocks. When the user later wants to verify the correctness of the data on the cloud he requests the cloud to provide the signatures over a specified blocks by the designated server storing the multiple redundant data and when the user receives these signature this should match with the tokens that were earlier generated by the user before uploading the data to the cloud.

The system also provides error recovery possibility by detecting the server which returned the in correct data. The system would then retrieve the data from the other server that are stored redundantly and transform the data the correct data may be recovered with a high probability. Thus the user would be able to request the block signatures from the cloud and on detecting a misbehaving server the data can be regenerated by using the erasure correction. The algorithm for data recovery is described in the following diagram [5]

1: **procedure**
% Assume the block corruptions have been detected among
% the specified r rows;
% Assume $s \leq k$ servers have been identified misbehaving
2: Download r rows of blocks from servers;
3: Treat s servers as erasures and recover the blocks.
4: Resend the recovered blocks to corresponding servers.
5: **end procedure**

Fig: 7 Error Recovery Algorithms

2.2.5 Preventing online cloud storage as Slack Space

The consumer based applications like Dropbox [7] have a huge customer base with millions of users and billions of file being stored. Though the system design allows very weak security and can be easily be manipulated to breach the privacy of unassuming customers and files uploaded could be easily be retrieved without much effort until recently. These services have the potential to be used as hidden channel to leak data stored on the system as described by Mulazzani et al [1]. Sensitive data could be uploaded to these services and as long as the users have the hash key to the file's chunks they could be retrieved by others limiting on the data that needs to be communicated through covert channels to just the file chunk keys. A more secure means of distributing sensitive information due to the slow removal process of the file chunks from the data storage services of Dropbox would be the malicious user could upload sensitive files to the service and later the when the upload process is over the user would need to identify the hash keys of the file chunks again and then delete the file from the service. Later colluding user would have to just provide the file hash key for each of the chunks and download them and merge them to receive the file [1].

As described in the paper [1] the storage system uploads the file in using a HTTPS protocol and the upload of files is actually a two-step process. After the file is uploaded to the file, a second request is sent to the servers to link the file chunk to the user, this would mean that user's may actually omit the second step in the upload process without much trouble if they could modify the uploader binary to skip the step. This would lead to scenario file chunks could be uploaded without any potential limit and be not linked to any particular user. Thus resulting in a situation where the service could be used as online version of slack space. The malicious users might user this in combination to live operating system CDs and would be effectively be able to use these hidden files without any trace being left out in the system about the files after the system is shut down.

The authors [1] also carried out the feasibility and study of the online storage space a means of slack space by performing a study of the duration for which file unlinked to any particular user resides on the server. They uploading randomly generated data file to the servers and then deleted these file immediately. These files where then attempted to be retrieved every twenty four hours by the authors and the following graph [1] was obtained based on the availability of those files.

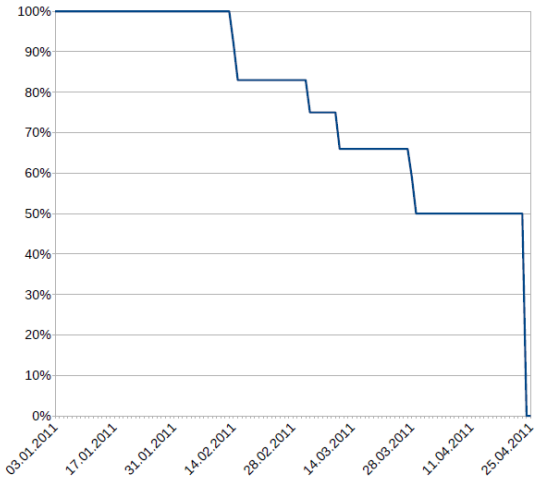


Fig: 8 File Availability after Deletion

From the figures it is quite apparent that about 50% of the files are still available even after 4 weeks. This

trend causes the possibility of use of storage space as slack if the system design is not implemented properly and these issues are dealt with during design time. They solution to these issues could be to a large extent mitigated by use of a secure data possession protocol is used. In order to do this a effective way would be use a challenge response scheme to verify the user as proposed in [1]. The sequence diagram [1] of the scheme is shown below:

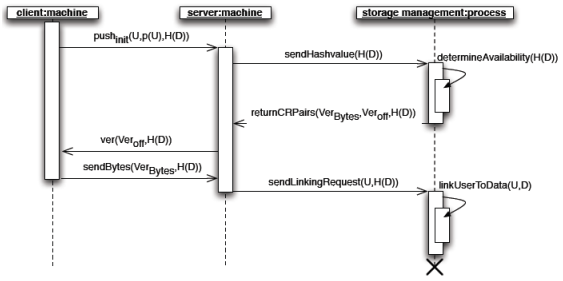


Fig: 9 Challenge Response Scheme

The initial data upload function $Push_{init}$ would send the user id, user identification token and the hash of the file data that needs to be uploaded based on the presence of the file hash, the data is then uploaded. When the user request for a file using Req function the server would ask for some randomly selected bytes from the verification array from the client and when the response matches the bytes that the server has the server would authorize the download of the file. To prevent usage of storage space as a slack space the provider should delete all the unlinked data file chunks immediately.

2.2.6 Distributing File Chunks across providers

All this while we were discussing about how properly secure the file on cloud storage service provider, or how to properly authenticate the user attempting for access. In all of these scenarios there could be situations in which a particular cloud service provider might be compromised and data of the user might be under a privacy threat. A good way to counter this would be encrypt the data as with most of the cloud service provide today. There could be legitimate cases in which a colluding cloud service provider due to monetary needs or other malicious intent might be able to have access to the data or patterns to the data stored in spite of the data being encrypted. Another possibility is that the cloud service provider might be obligated to provide the data present in the cloud to the authorities under a subpoena.

With respect to these scenarios I would like to propose a solution in which the risk is spread among various cloud service providers. At the very least this would require more effort on behalf of the malicious attackers since now to corrupt or steal a user's data more than one cloud storage service providers have to be dealt with. The solution should be transparent enough for the user to randomly select cloud storage service provider for each of the file he uploads. The data never leaves the client's machine in the clear and the client and no one else is aware of the symmetric key to encrypt the data that he is uploading.

Threat Model: The system that I propose assumes that the client is completely trustable and that the symmetric keys that are used for encrypting are known only to the respective user.

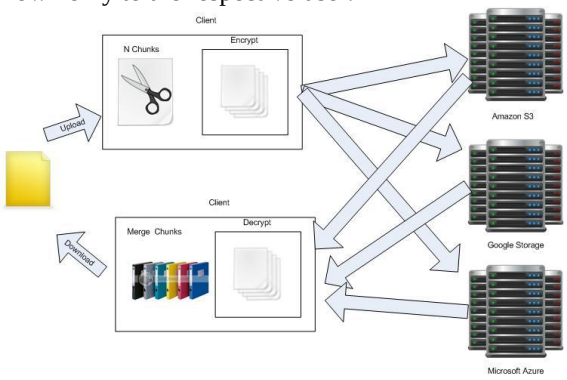


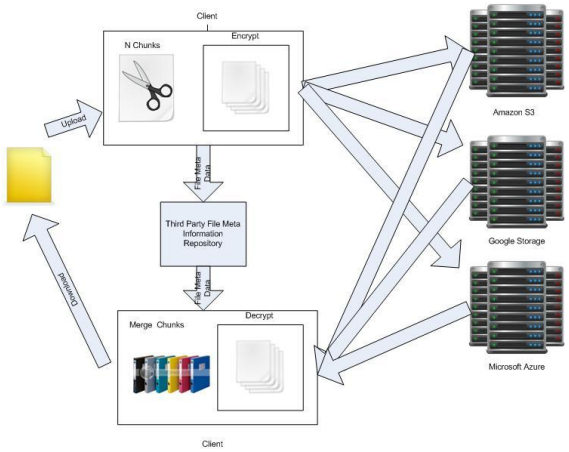
Fig: 10 Basic Architecture

The channel that is used to upload and download the files is assumed to be secure by use of suitable channels such as HTTPS. The cloud service providers are assumed here to be untrusted and may compromise the privacy of the customer by exposing the data to third parties or reading it themselves.

Figure 10 describes a basic architecture of the system that is proposed. The files that need to be uploaded are broken into chunks of equal size based on the number of cloud storage systems to which this file needs to be sent. Once the file that needs to be downloaded is indicated the respective file chunks from different service providers are downloaded and decrypted locally on the client and then merged to form the file. In this scenario the system acts as a means to back up files to the cloud certainly a slight modification of the system with an untrusted third party could result in a system which could potentially sync files between various systems of the client when he has set the system up. The modified architecture is shown below. Earlier in the backing up the data scenario the client maintains the Meta data information regarding the file's chunks and the service with which each of the clients is stored at and using this information to download the file. If a third party system is set up which would provide web services to the linked clients such that any client that provides the correct potential are provided with these meta data information could then download the file chunks from the respective cloud services and decrypt and merge the chunks into a single file by themselves.

Since this third party is also untrusted the client does not pass the file encryption keys to this service provider, at the same time the file meta data information are secured by using a user

authentication scheme using the web services API exposed by the service provider.



Thus the user would be in total control of the data and at no point of time any of the untrusted third parties including the cloud storage service provider has knowledge of the data that is stored. The solution has potential attack possibilities in terms of the client itself being compromised but the assumption here is that the client code would be an open and would be openly audited for issues and vulnerabilities and would result in a robust piece of code with minimum vulnerabilities. As for the third party service provider they could use their own proprietary systems to store the file Meta information, but this information could be encrypted by the client and then sent to the provider in order to avoid potential threats. As for the cloud service provider at no point of the scheme do they have any information regarding the type of file or whether the files are complete or chunks, thus they would not be able to provide any feasible data in case of subpoenas by the authorities thus ensuring appropriate privacy protection.

3. Conclusion

In this discussion we saw various solutions on how to ensure that data stored in the cloud is not maligned or corrupted by the service providers or other attack agents using various types of challenge response schemes in order to occasionally test the service provider for quality of data provided and ensuring data is correct. Another solution provides a technique to enforce security on the service providers by using provenance labels so that the clients or consumers are assured that they get the correct service they are paying for and thus ensuring maximum security for their data. We have also seen schemes to ensure the history of data is maintained during cloud transactions so that issues or origin of data could be identified easily and provide means for detecting and identify security violations. There is possibility of the online cloud storage system to be used as an online slack space for malicious use or distribution of malicious data without consequences and we saw how this could be prevented by using upload and downloader verification in order to properly authenticate the user client being used in these services. Finally I myself propose a solution to take care of privacy and security threats by distributing the data among various service providers thus reducing the risk by storing all your data with just one service provider.

Securing a cloud service and providing privacy protection to customer and his data can be quite a daunting task, it would require a substantial effort on behalf of the cloud service provider and the industry in general to implement some of the techniques that have been explained here. Although as indicated by another research by Chen et al providing merely strong encryption would not suffice for the lack of trust in cloud service providers and most of the time it would not make an economic sense. To a large extent some of the onus lies with the service providers to live up to their reputation by implementing various features to ensure security and privacy in the services they provide.

4. Acknowledgement

I would like to thank Prof. Konstantin Beznosov for providing me with an opportunity to carry out this research as part of the curriculum of ECE 571B course and for his inputs that he provided on reviewing my abstract. I would also like to thank the UBC library system for providing me access to a plethora of research papers that I have been looking for in this topic.

5. References

[1] Martin Mulazzani, Sebastian Schrittwieser, Manuel Leithner, Markus Huber, and Edgar Weippl, *SBA Research*, Dark Clouds on the Horizon: Using Cloud Storage as Attack Vector and Online Slack Space , 20th USENIX Security Symposium, 2011

[2] Prince Mahajan, Srinath Setty, Sangmin Lee, Allen Clement, Lorenzo Alvisi, Mike Dahlin, and Michael Walfish, *The University of Texas at Austin*, Depot: Cloud storage with minimal trust, 9th USENIX Symposium on Operating System Design and Implementation, 2010

[3] Raluca Ada Popa, *MIT;* Jacob R. Lorch, David Molnar, Helen J. Wang, and Li Zhuang, *Microsoft Research*, Enabling Security in Cloud Storage SLAs with CloudProof, USENIX Annual Technical Conference, 2011

[4] Kiran-Kumar Muniswamy-Reddy, Peter Macko, and Margo Seltzer, *Harvard School of Engineering and Applied Sciences*, Provenance for the Cloud, 8th USENIX Conference on File and Storage Technologies, 2010

[5] Cong Wang, Qian Wang, Kui Ren, Wenjing Lou, Dept. of ECE, *Illinois Inst. of Technology*, Ensuring data storage security in cloud computing, 17th International Workshop on Quality of Service, 2009

[6] Giuseppe Ateniese, *The Johns Hopkins University*; Roberto Di *Pietro, Universitat Rovira i Virgili*; Luigi V. Mancini, *Università di Roma "La Sapienza"*; Gene Tsudik, *University of California Irvine*, Scalable and efficient provable data possession, SecureComm 2008 Proceedings of the 4th

International Conference on Security and Privacy in Communication Networks
[7] Dropbox.com, *The Dropbox Blog*, June 2011, Online at http://blog.dropbox.com/?p=821
[8] SpiderOak.com, *Why Spideroak?* February 2012, Online at https://spideroak.com/whyspideroak#privacy
[9] N. Belaramani, M. Dahlin, L. Gao, A. Nayate, A. Venkataramani, P. Yalagandula, and J. Zheng. PRACTI replication. In NSDI, 2006.
[10] V. Ramasubramanian, T. Rodeheffer, D. B. Terry, M. Walraed-Sullivan, T. Wobber, C. C. Marshall, and A. Vahdat. Cimbiosys: A platform for content-based partial replication. NSDI, 2009.
[11] P. Reiher, J. Heidemann, D. Ratner, G. Skinner, and G. Popek. Resolving, File Conflicts in the Ficus File System. USENIX Summer, 1994.
[12] D. B. Terry, M. M. Theimer, K. Petersen, A. J. Demers, M. J. Spreitzer, and C. H. Hauser. Managing update conflicts in Bayou, a weakly connected replicated storage system. In SOSP, 1995.
[13] B.-G. Chun, P. Maniatis, S. Shenker, and J. Kubiatowicz. Attested Append-Only Memory: Making Adversaries Stick to their Word. SOSP, 2007.
[14] D. Levin, J. R. Douceur, J. R. Lorch, and T. Moscibroda. TrInc: small trusted hardware for large distributed systems. In NSDI, 2009.
[15] J. Li and D. Mazi`eres. Beyond one-third faulty replicas in Byzantine fault tolerant systems. NSDI, 2007.
[16] C. Cachin, I. Keidar, and A. Shraer. Fail-Aware Untrusted Storage. DSN, 2009.
[17] J. Li, M. Krohn, D. Mazi`eres, and D. Shasha. Secure untrusted data repository (SUNDR). OSDI, 2004.
[18] Gray, J., Slutz, D., Szalay, A., Thakar, A., Vandenberg, J., Kunszt, P., AND Stoughton, C. Data Mining the SDSS SkyServer Database. Research Report MSR-TR-2002-01, Microsoft Research, January 2002.
[19] Muniswamy-Reddy, K.-K., Holland, D. A., Braun, U., Seltzer, M. Provenance-aware storage systems. 2006 USENIX Annual Technical Conference.
[20] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In ACM CCS'07
[21] A. Juels and B. Kaliski. PORs: Proofs of retrievability for large files. In ACM CCS'07
[22] Blaze, M. A cryptographic file system for unix. In ACM CCS (1993).
[23] Bindel, D., Chew, M., And Wells, C. Extended cryptographic filesystem. In Unpublished manuscript (1999).
[24] E.-J. Goh, H. Shacham, N. M., AND Boneh, D. Sirius:Securing remote unstrusted storage. In NDSS (2003).
[25] Kallahalla, M., Riedel, E., Swaminathan, R., Wang, Q., AND Fu, K. Plutus: Scalable Secure File Sharing on Untrusted Storage. In USENIX FAST (2003).
[26] Li, J., Krohn, M., Maziered, D., AND Shasha, D. Sundr: Secure untrusted data repository. In OSDI (2004).
[27] Boneh, D., Gentry, C., AND Waters, B. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. Lecture Notes in Computer Science (2005).
[28] Fiat, A., AND Naor, M. Broadcast encryption. Proc. of Crypto (1993).
[29] A. Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, pp. 584–597, 2007
[30] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. of Asiacrypt '08, Dec. 2008.
[31] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Cryptology ePrint Archive, Report 2008/175, 2008, http://eprint.iacr.org/.
[32] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. Of CCS '07, pp. 598–609, 2007.
[33] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. of SecureComm '08, pp. 1–10, 2008.
[34] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. of ICDCS '06, pp. 12–12, 2006.
[35] M. Lillibridge, S. Elnikety, A. Birrell, M. Burrows, and M. Isard, "A Cooperative Internet Backup Scheme," Proc. of the 2003 USENIX Annual Technical Conference (General Track), pp. 29–41, 2003.
[36] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Cryptology ePrint Archive, Report 2008/489, 2008, http://eprint.iacr.org/.
[37] J. Hendricks, G. Ganger, and M. Reiter, "Verifying Distributed Erasurecoded Data," *Proc. 26th ACM Symposium on Principles of Distributed Computing*, pp. 139–146, 2007.
[38] T. S. J. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," *Proc. of ICDCS '06*, pp. 12–12, 2006.
[39] Oleshchuk, V.A.; Koien, G.M.; , "Security and privacy in the cloud a long-term view," Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE), 2011 2nd International Conference on , vol., no., pp.1-5, Feb. 28 2011-March 3 2011
[40] Y. Chen and R. Sion, On securing untrusted clouds with cryptography, In Proceedings of the 9th annual ACM workshop on Privacy in the electronicsociety, pp.109–114, ACM WPES '10, 2010

6. Appendix

Amazon EC2: Amazon's product providing computing service with features for robust scaling of the applications.

Amazon S3: Amazon's Cloud Storage solution using simple key value storage system.

Cloud: An umbrella term using for referring to the global network and in this case the Internet.

Cloud Broker: As the term suggests this is an entity that maintains relationships with various cloud service provider and acts as a intermediary between the clients consuming the cloud services and help them audit the services.

Cloud Probability: The probability of an application and data moved from another cloud service provider working seamlessly with the service that it has been moved to.

HaaS: Hardware as a service business model.

IaaS: A business model under which service providers are able to provide a virtual representation of various hardware such as servers and network components as a service on demand over the internet.

PaaS: Platform as a service, where software components such as an operating system with the associated development and deployment software are provided as a on demand service in a virtualized environment over the internet.

SaaS: A virtualized means of providing applications over the internet and the ability to use and get charged based on the consumption levels of the consumers.