# A Secured Distributed and Data Fragmentation Model for Cloud Storage

Lixuan Wang, Lifang Liu, Shenling Liu, Dong Chen, Yujiao Chen
School of Computer
National University of Defense Technology
Changsha, China
Wlxeva88720@126.com

*Abstract*—**The increasing popularity of cloud service is leading people to concentrate more on cloud storage than traditional storage. However, cloud storage confronts many challenges, especially, the security of the out-sourced data(the data that is not stored/retrieved from the tenants' own servers). Security not only can keep the data from attacking but also can recover the original data after attack efficiently. Thus, to address the security issue, we proposed a new distributed and data fragmentation model of cloud storage named DDFM (Distributed and Data Fragmentation Model). DDFM aims to provide tenants a secured and integrated cloud storage service with layer-to-layer protection strategy. The layer-to-layer protection strategy of our model includes three main algorithms: the Authentication and Authorization Management Algorithm based on OpenID and OAuth, the Data Fragment Algorithm based on Granular Computing and the Haystack File Storage Algorithm. Considering tenants' security requirement our model DDFM based on these algorithms provided a better decision of cloud storage architecture for our tenants. Furthermore, DDFM can defense most of the network threats and provide a secured way for the third-party applications to access sensitive information that stored on the cloud storage.**

***Keywords-cloud storage, secured, OpenID, OAuth, data fragmentation, Haystack***

## I. INTRODUCTION *(HEADING 1)*

The major organizations are looking for a demand storage option because the data storage capacity are increased double year by year[1]. The popularity of cloud service is leading organizations to concentrate on cloud storage, such as Amazon's Elastic Compute Cloud (EC2). Furthermore, one of the main challenges of cloud storage nowadays is security.

In a public cloud, tenants may feel insecurity about participating in it, because they can delegate system administration to cloud providers, but this also means that administration and operations are not controlled by tenants themselves. An IDC survey on Cloud/On-Demand showed that more than 70% of potential cloud tenants regard security as a major reason against adopting cloud [4].

Recently, KevinD.Bowers, Christian Cachin and Hakim Weatherspoon presented many ideas of Cloud Storage. Kevin D.Bowers introduced HAIL(High-Availability and Integrity Layer),a distributed cryptographic system. It can ensure a stored file's intactness and retrievability[2].Whereas, it only provided assurance for static files, not for the dynamic data. Christian Cachin presented to solve the problem of data integrity and consistency as well through using cryptographic tools. And Hakim Weatherspoon introduced the Antiquity, a wide-area distributed storage system, it combined a secured append-only log interface to maintain data integrity, consistency and durability [3]. Unfortunately, Antiquity cannot prevent the occurrence of security issues. To nip the security problems in the bud is better than to compensate afterwards.

Therefore, the existing ideas of cloud storage are mainly concentrated on one aspect of security: performance, integrity, back-up and etc. But the system still have security problems. Whereas, cloud-tenants focus more on security of the whole process, that is, the security of data storage and data transfer. So we are seeking a more efficient and secured cloud storage model.

In this paper, to address the security issue, we proposed a new distributed and data fragmentation model of cloud storage named DDFM (Distributed and Data Fragmentation Model). In addition, providing better security of data through layer-to-layer protection strategy as well as ensuring the original benefits of cloud storage, such as data flexibility, cost reduction and highly automated, can be achieved by DDFM. The layer-to-layer strategy of our model includes three main algorithms: the Authentication and Authorization Management Algorithm based on OpenID and OAuth, the Data Fragment Algorithm based on Granular Computing and the Haystack File Storage Algorithm.

We organize the remainder of this paper as follows.

To begin with we briefly provides the background of cloud storage and its architecture in section II, followed by our design of Distributed and Data Fragmentation Model (DDFM) which is a high efficiency and secured model of cloud storage discussed in section III. In section IV, we analyzes the security threats of cloud storage and the security of DDFM. The last part is the conclusion of this paper.

## II. BACKGROUND

### *A. Cloud Storage*

Cloud storage is somehow the same as cloud computing, and is an extension of cloud computing. It consolidates all storage devices like cluster application, grid technology and distributed file system.

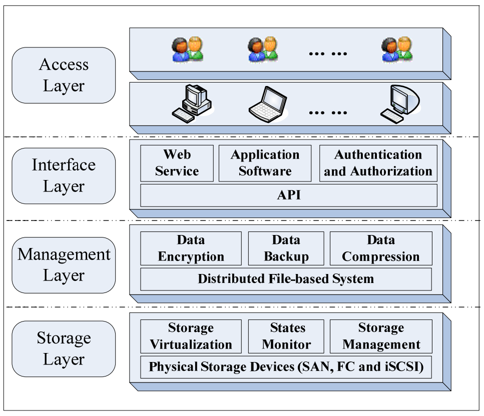

Figure 1. The Architecture of Cloud Storage:

Cloud Storage architecture contains four layers generically. As figure 1 shown, they are Access Layer, Interface Layer, Management Layer and Storage Layer. Respectively Access Layer provides users to access the storage. Behind the access layer there's a front end that exports an API named Interface Layer. In traditional storage systems, this API is the SCSI protocol, but in cloud storage service, these protocols are evolving[1]. This layer includes Web service front ends, application software front ends, and more traditional front ends such as authentication and authorization front ends. Behind the front end is a layer named Management Layer. This layer implements a variety of features, such as data encryption, data backup, data compression and distributed file-based system. Finally, the back end is Storage Layer, it implements specific features such as storage virtualization, states monitor and includes the physical storage devices, like SAN, FC and iSCSI.

In this paper, we proposed a layer-to-layer security strategy for different layers based on this storage architecture.

## B. OpenID

OpenID is an open and decentralized authentication standard describes how users be allowed to consolidate their digital identities. OpenID users can choose a trustworthy OpenID server randomly to register their OpenID account. OpenID account is not a simple pair of username and password, is an URL that cannot be captured by attackers easily. In OpenID mechanism, three parts are involved: the OpenID provider (OP), the service provider which is also called Relying Party (RP) and the end user. The typical authentication flow of OpenID is described in figure 2.

1. End user inputs OpenID and submits it to RP;
2. RP normalizes user's OpenID and identifies the user, then redirects the user to OP;
3. User inputs password for authentication;
4. OP authenticates user by OpenID account, if the authentication is successful, OP will redirect user to RP.

## C. OAuth

OAuth is an open authorization standard that describes how users are allowed to share their sensitive resource stored on storage providers with other third-party applications without showing their access permissions or the full extent of their resource, supplying access tokens instead. It provided a secured communication mechanism because the access permissions would not be divulged.

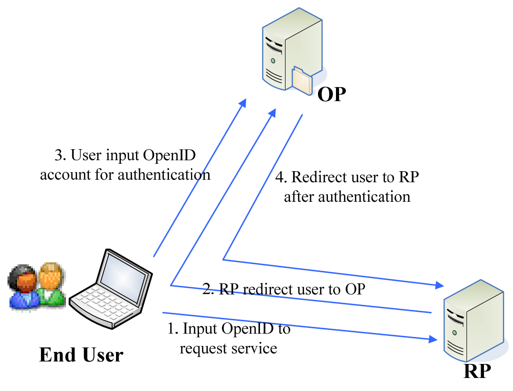
OP
3. User input OpenID account for authentication
4. Redirect user to RP after authentication
2. RP redirect user to OP
1. Input OpenID to request service
End User
RP

Figure 2. Authentication Flow of OpenID

## D. Data Fragment Algorithm based on Granular Computing

The data fragment algorithm based on granular computing can give the data localization a suitable priority. And tenants can make an adjustment in the global relation partition's coarse and fine dynamically, to obtained a rational sate for the data fragments. Meanwhile, it will reduce the network flow, decrease the system cost, control the number of data fragments and decrease the data joint workload, thus improving the system efficiency [5].

We used this algorithm in Management Layer to implement the data encryption.

## E. Haystack File Storage

Haystack is a file storage system optimized for Facebook's photo application. It includes three core components, Haystack Directory, Haystack Store and Haystack Cache. It provided a less expensive, higher performance and expandability file storage system, introduced by Facebook. In our DDFM model, we applied the Haystack file storage algorithm in Storage Layer.

## III. MODEL

As we known before, the cloud storage architecture has four layers: Access Layer, Interface Layer, Management Layer and Storage Layer. We need to construct a layer-to-layer security insurance, and provide different strategies for each layer, so that the full range of protection from data transfer to data storage can be achieved. Our model DDFM combined the Information Diffusion, Distributed Storage Management and Data Recovery technologies and realized the secured data storage and management.

Our model includes three main algorithms, the Authentication and Authorization Management Algorithm based on OpenID and OAuth named DAAM; the Data Fragment Algorithm based on Granular Computing, GDFS; and the Haystack File Storage Algorithm. As figure 3 shows.

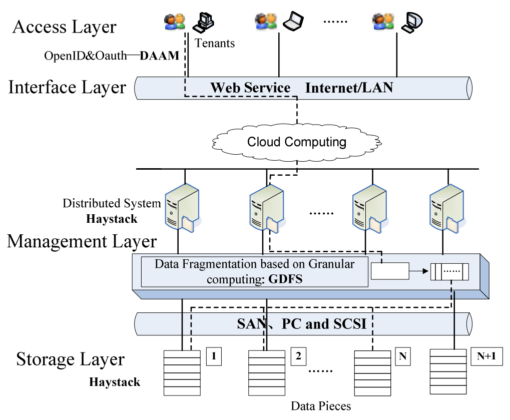

Figure 3. The structure of the secured DDFM model.

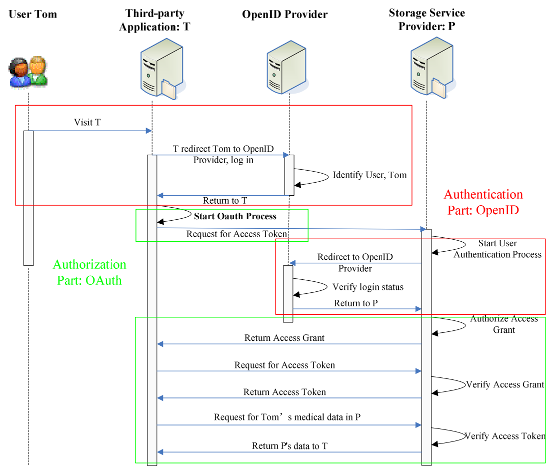

Figure 4. The Process of DAAM

### A. *Design of Access Layer and Interface Layer*

In this part, our model DDFM uses a new decentralized authentication and authorization mechanism based on OpenID and OAuth technologies to enhance the security of Access Layer and Interface Layer. This new mechanism named DAAM (Decentralized Authentication and Authorization Mechanism), provides a flexible, general and secured authentication mechanism for multiple service providers in cloud storage and an declarative authorization mechanism based on tokens. So users can authorize the third-party applications to access their shared resource (data pieces) without sharing their access permits.

Cloud storage includes more than one service providers, so the proliferation of user ID is a big problem of multiple identities for the same user in a multiple service providers' model. A simple way to overcome this problem is by adopting OpenID mechanism. OpenID is a decentralized authentication mechanism (There is no central OpenID server that holds all the security information in the universe.); it works by registering one primary OpenID account as the key to the authentication of a single end user with multiple service providers [10].

Cloud storage can provide tenants a secured place to store their sensitive information. Furthermore, tenants also need to share those information with other third-party applications. During the sharing process, security problem of authorization is much more important. Therefore, to seek an efficient and secured authorization mechanism is critical and this can be achieved by the authorization technology OAuth.

With DAAM, tenants can authorize the third-party applications to access their shared sensitive information that stored on multiple service providers without sharing their access permits.

Figure 4 describes the whole process of DAAM, including authentication and authorization flows.

Let's see our tenant Tom's example, Tom stored his medical data in storage service provider P. The third-party application T wants to get Tom's medical information from P. So the whole process of DAAM is as follows:

1. Tom should be authenticated firstly. When Tom visits the third-party application T, DAAM will redirect him to the OpenID provider (OP, where Tom registered his account) to verify his account. Tom will log into the OpenID provider with his specific URL.

2. Second, DAAM will return Tom to T if the authentication successes.

3. After that, T will guide Tom to P to take his P's data which will send a request of access token, such as a question like, "T wants to take your P's data. Is that cool?". The answer "Yes" is the authorization from Tom. With the authorization, P will return an Access Grant to T.

4. After the Access Grant received by T, T will send a request for Access Token to P.

5. P will verify the Access Grant and return the Access Token to T.

6. With the Access Token, T finally requests the data from P.

7. If the Access Token is correct, P will send back Tom's medical data to T without sharing the access permits.

With OpenID and OAuth mechanisms, our tenant Tom doesn't need to re-enter his profile details and log into P again, so Tom's profile details will not be divulged. Moreover, the third-party application T can access P and take P's data directly without sharing access permits. So the attackers cannot break into P with the access permits to steal Tom's sensitive medical information.

Therefore, in our Access Layer and Interface Layer, DAAM can provide tenants a secured and safe authentication and authorization mechanism and enhance the tenants' confidence of our cloud storage service.

### B. *Design of Management Layer*

Privacy and security of data are the key points of our model DDFM, thus, we will pay close attention to the data encryption strategy in Management Layer.

As the regular data encryption strategy such as DES, IDEA and RSA are low efficiency and high cost, so the most popular strategy of data encryption nowadays is data fragmentation, which is a useful way to break sensitive associations between information. At first, we should clarify what is sensitive information. Sensitive is their association with others. For an instance, in hospital the list of patients or the list of medicines that the patients have is not sensitive information, but the association of specific medicine to individual patient is sensitive. Hence, we aim to seek a perfect strategy to protect our tenants' sensitive information.

Actually, there are many types of data fragmentation strategies, such as horizontal fragmentation, vertical fragmentation, hash fragmentation and mixed fragmentation. But they are not suitable for semi-structure data. The vertical fragmentation only produces two data pieces during each data fragment procedure and it's difficult to find a perfect hash algorithm in hash fragmentation. Thus, the traditional data fragmentation strategies are either high-cost or difficult to implement.

Therefore, in the Management Layer, the data fragmentation strategy that we use is Granular Data Fragmentation Strategy (GDFS).

GDFS is a data fragment algorithm based on granular computing, which can give the data localization a suitable priority. Tenants can make an adjustment in the global relation partition's coarse and fine dynamically to obtain a rational sate for the data fragments. Meanwhile, it will reduce the network flow, the system cost and the data joint workload by control the number of data fragments. Thus, GDFS improves the system efficiency [5].

GDFS divided the data into several pieces, and they have redundancy with each other so that the data pieces are not visible and cannot be recognized by other systems. For every single piece of data, it's meaningless even if the data piece has been picked up by attackers during the data transfer or scanned by Trojan. Because the attackers or the Trojan only picked up or scanned some of the data pieces, not the full version of the integrated data and don't know the associations, we can guarantee that the sensitive information will not be divulged or proliferated. Besides, the data pieces would be stored into different storage service providers in different locations (it will be introduced in Storage Layer). If one or two of the storage service providers failed or crashed, the rest of providers also can comprise the whole data because of the redundancy of those data pieces. Therefore, the data fragmentation is high fault-tolerance to protect sensitive data and it can solve the problem of single point of failure. Thus, the granular data fragmentation strategy will enhance the security of our DDFM model during the Management Layer.

## C. Design of Storage Layer

Storage Layer is the keystone of our model DDFM. In this layer, the requirements of less expensive, higher performance, low latency and expandability can be achieved by Haystack. Our cloud storage service providers process billions of tiny data pieces a day. All the tiny data pieces which were fragmented by our data fragmentation strategy GDFS in the management layer will be stored into different storage service providers in different locations and will have redundancy with each other. So this distributed file storage algorithm Haystack can improve the ability of fault-tolerance, recovery and avoid Single Point of Failure.

Otherwise, multiple service providers will exchange and share their data pieces with each other to reconstruct an integrated data for their tenants. Because of the huge amount of small data pieces, the excessive number of disk operations is the bottleneck. And the main reason of high disk operations is the metadata lookups. Thus, we introduced Haystack as our storage algorithm in this part. Haystack is a storage system optimized for Facebook's photo application which can perform all metadata lookups in main memory[6].

There are three core components in Haystack, we call them Haystack elided. They are Haystack Directory, Haystack Store and Haystack Cache.

In Haystack Store, for example, if we have a 10 terabytes capacity data of a service provider, we can put it into 100 physical volumes each of which provides 100 gigabytes of storage. Moreover, we further group physical volumes on different service providers into logical volumes. This redundancy can mitigate data loss due to hard drive failures, disk controller bugs, etc.[6]

In Haystack Directory, it contains the logical volume to physical volume mapping, such as which logical volume each data piece belongs to and which is the free space in the logical volumes.

Haystack Cache, as our internal CDN, provides a store for our most popular data to reduce the I/O cost. Figure 5 describes the Read and Upload service flow of Haystack.

Figure 5 depicts the "Read" service flow of Haystack.

*Read:*

When a tenant visits a page (step1), the web server uses the Haystack Directory (step 2) to construct an URL for each data piece and return it to browser. The URL includes several pieces of information as figure 6 shows.

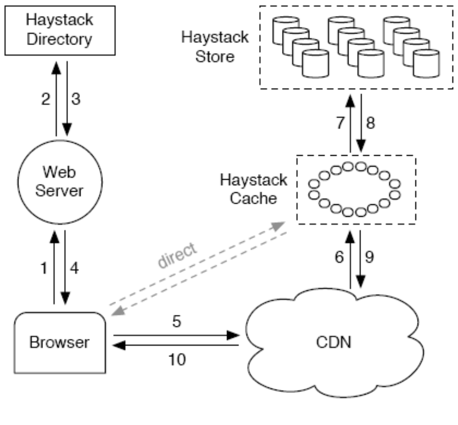

Figure 5. Service Flow of Read [6]

http://⟨CDN⟩/⟨Cache⟩/⟨Machine id⟩/⟨Logical volume, Photo⟩

Figure 6. The format of URL[6]

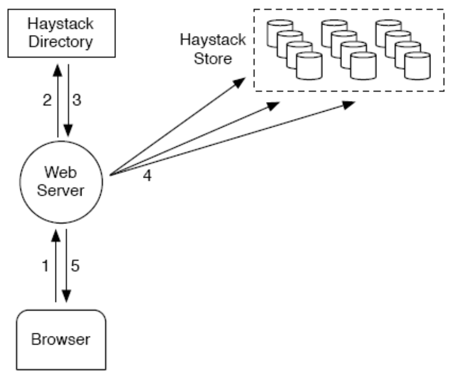

Figure 7. Service Flow of Upload [6]

Then Haystack will direct the browser to CDN (step3, 4, 5) and lookup the data internally. If CDN cannot find the data that we need, it will contact the Haystack Cache (step6).The Cache will do a similar lookup to find the data and, on a miss, Haystack will go directly to find on the specified Haystack Store (step7).After reading the data successfully, Haystack will go back to the browser (step 8, 9, 10) and give the data to our tenants. Otherwise, if the URL doesn't have the CDN information, browser can access the Haystack Cache directly (see the dotted line) and complete the steps as figure 7 shows.

*Upload*:

When Haystack stores a data piece on a logical volume, the photo is written to all corresponding physical volumes [6]. For instance, Figure 8 illustrates the "Upload" flow of Haystack. When a tenant uploads a data, he first sends the data to web server (step1). Then the web server requests a write-enabled logical volume from the Haystack Directory (step2). When our tenant gets the unique id for the data (step3), he will upload his data into the physical volumes that mapped to the assigned logical volume (step4, 5).

Thus, we chose Haystack in the Storage Layer to improve the ability of fault-tolerance and recovery, and avoid the Single Point of Failure.

## IV. SECURITY ANALYSIS

### A. Network Attack

Cloud storage runs on a network structure so they are open to network type attacks, such as the network sniffing. The sniffer can capture sensitive information if unencrypted such as passwords or other web service related security strategies. For example, the UDDI (Universal Description Discovery and Integrity), SOAP (Simple Object Access Protocol) and WSDL (Web Service Description Language) files[9]. Another network type of attack is loss of governance such as the vicious deletion or interpolation.

In Access Layer and Interface Layer of our model DDFM, network sniffing cannot succeed because our OpenID authentication mechanism use specific URL created by OpenID provider instead of simple username and password. Otherwise, vicious deletion and interpolation are also unsuccessful. Because the authorization mechanism OAuth provides tenants a secured way to share their data by asking for access tokens instead of the tenants' permissions. So attackers cannot vicious delete or interpolate tenants' data without the access permissions.

### B. Un-controllability

Flexibility and simplicity are the benefits of cloud storage, but they are double-edged swords. For the tenants' outsourced data, tenants can administrate the cloud providers easily and flexibly also means that administration and operations are not controlled by the tenant.

Whereas, in our model DDFM, administrations and operations must controlled by tenants whom with legal authorization by OAuth. We assume that one cloud storage provider has been attacked, but the attacker still cannot get the rest data pieces stored in other providers, because he doesn't have the access permissions that the tenants provided. So that the attacker cannot get all the data pieces for reconstructing the original data unless all of the providers are attacked at the same time, this is almost impossible.

### C. Single Point of Failure

The Single Point of Failure could affect the data availability and consistency and occur if a cloud service provider failed or crashed, which makes it hard for tenant to retrieve his stored data from the provider [8]. For example, a tenant Tom, he stored his medical information (sensitive information) on a service provider (CSP). Tom can retrieve his medical information from CSP which he has a contract with. If a failure occurs at the CSP, all Tom's medical information which was stored on CSP will be lost and cannot be retrieved. This is the Single Point of Failure.

Our DDFM's data fragmentation strategy GDFS and storage algorithm Haystack allow tenants sharing their sensitive data pieces that stored on different storage providers, to avoid the single point of failure. If the service interruption occurs on one of the cloud storage providers because of the disasters, such as power failure, earthquake, fire and so on. Then other providers still can reconstruct the original integrity data for tenants because the data pieces that stored on those different providers have redundancy. Therefore, DDFM has high fault-tolerance and recovery ability.

Thus it can be seen, the security issue runs through all layers of cloud storage architecture. Our distributed and data fragmentation modelDDFM, is based on a layer-to-layer strategy, can defense most of the security threatssuccessfully, such as those threats we analyzed above:the network attacks, single point of failure and un-controllability.Therefore, DDFM can provide a secured cloud storage service for tenants.

## V. CONCLUSION

In this paper, we proposed a distributed and data fragmentation model (DDFM) of cloud storage, which seeks to provide tenants a secured and integrated cloud storage service with layer-to-layer protection strategy. The layer-to-layer strategy of our model includes three main algorithms, the Authentication and Authorization Management Algorithm based on OpenID and OAuth, the Data Fragment Algorithm based on Granular Computing and the Haystack File Storage Algorithm. By using the decentralized authentication and authorization management mechanism DAAM, our model can provide a secured way for other third-party applications to access the sensitive information that stored on cloud storage service providers; by dividing and distributing data with the data fragment algorithm based on granular computing named GDFS, our model can guarantee that sensitive information will not be divulged or proliferated; and Haystack provided a less expensive, higher performing and expandability file storage system to enhance the security of our model DDFM.

## ACKNOWLEDGMENT

R.B.G thanks for the supports from Fund No.KJ-12-06.

## REFERENCES

[1] Talasila Sasidhar, Pavan Kumar Illa, and Subrahmanyam Kodukula,“A Generalized Cloud Storage Architecture with Backup Technology for any Cloud Storage Providers”, International journal of computer application, ISSN:2250-1797, Issue2, Vol.2, April 2012.

[2] Kevin D.Bowers, Ari Juels, and AlinaOprea,“HAIL:A High-Availability and Integrity Layer for Cloud Storage”, Proceedings of the ACM Conference on Computer and Communications Security, p 187-198, 2009.

[3] Hakim Weatherspoon, Patrick Eaton, Byung-Gon Chun, and John Kubiatowicz, “Antiguity: exploiting a secure log for wide-area distributed storage,” EuroSys ’07 Proceedings of the 2nd ACM SIGOPS, June 2007, pp. 371-384.

[4] Zhaoji Lin, Ping Lu, Shengmei Luo, Feng Gao, and Jianyong Chen,“An On-Demand Security Mechanism for Cloud-Based Telecommunications Service”, ZTE Communications, Vol.9 No.1, March 2011.

[5] Runxiu Wu, Shuixiu Wu, and Qing Liu, “Data fragment algorithm based on granular compuring”, Journal of Computer Applications, June 2007, Vol. 27 No.6, pp 1388-1391.

[6] Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel, “Finding a needle in Haystack: facebook’s photo storage”, Proceedings of the 9th USENIX conference on Operating system design and implementation, Article No.1-8.

[7] Eric Eldon,“Single sign-on service OpenID getting more usage”, Available:http://venturebeat.com/2009/04/14/single-sign-on-service-openid-getting-more-usage/.

[8] Yashaswi Singh, Farah Kandah, Weiyi Zhang , “A Secured Cost-effective Multi-Cloud Storage in Cloud Computing”, 2011 IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS 2011, p 619-624, 2011.

[9] Danish Jamil, Hassan Zaki,“Cloud Computing Security”, International Journal of Engineering Science and Technology(IJEST), ISSN : 0975-5462,Vol. 3 No. 4 April 2011.

[10] Anu Gopalakrishnan,“Cloud Computing Identity Management”, SETLabs Briefings, Vol. 7 No.7 2009.

[11] Valentina Ciriani, Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati, “Fragmentation and Encryption to Enforce Privacy in Data Storage”. Springer-Verlag, Lecture notes in Computer Science, 2008.