Information Security through Normalization in Cloud Computing

A.Sabina Parveen, C.R.Suganya

Syed Ammal Engineering college,Ramnathapuram,Tamil Nadu,India

Abstract- Data confidentiality is one of the pressing challenges in the ongoing research in Cloud computing. Hosting confidential business data at a Cloud Service Provider (CSP) requires the transfer of control over the data to a semi-trusted external service provider. Existing solutions to protect the data mainly rely on cryptographic techniques. However, these cryptographic techniques add computational overhead, in particular when the data is distributed among multiple CSP servers. This paper proposes a fragmentation technique which efficiently stores the data on CSP servers using the minimum possible amount of encryption. The fragmentation procedure is applied to a relational database where the tables are treated as independent fragments. This fragmentation and distribution approach reduces the trust expectancies towards the external service providers and thus improves privacy and confidentiality.

Index Terms- Cloud Computing, Data confidentiality, Privacy-preserving, Data fragmentation.

I. INTRODUCTION

Nowadays, Cloud computing , is the most promising novel technological approach for outsourcing IT services. It offers remarkable features such as scalability, virtualization, almost unlimited amounts of computational resources, pay per use, utility computing, data and process outsourcing etc. Providers like Amazon , Google , IBM , Microsoft and Oracle continuously focus their resources to evolve, improve, and extend their provided services of Cloud computing technologies . Services like on demand usage, pay as you use, and scalability proved to be highly cost effective and reliable and therefore caught the attention of many enterprises. However, even state-of-the-art technologies sometimes fail to provide an adequate level of privacy and confidentiality as service providers may be tempted to exploit the trust of customers and profit from passing private data to other parties. Moving sensitive data at CSPs also means that these providers have to be trusted that the data is protected adequately against external attacks . In the last couple of years, significant data breaches involving theft of customer credit card numbers, personal records, or other customer records indicated that even the larger organizations are prone to unintended data exposure . Preserving data security is a challenging task because of numerous issues: i) domain specific restrictions, ii) proper implementation of suitable security mechanisms, iii) variation of security measurements among the providers, iv) data aggregation, v) efficient data manipulation, vi) locality restrictions, vii) law restrictions, viii) domain limitations, and ix) service level security issues. In Cloud databases, data is stored on multiple dynamic virtual servers across the Cloud. Before being distributed to multiple servers, the data is fragmented. However, usage of pure cryptographic techniques to protect the data can impose heavy computational overheads and also raises key management concerns from the data owner's and provider's point of view. Depending on the application domain, e.g. data used in the IT domain, the user requires the data to be accessible in real time. Therefore, the main focus of our research is directed to preserve confidentiality and privacy while providing efficient accessibility and manageability. Currently, there are no adequate technical solutions for handling data fragmentation and encryption in the Cloud .

Existing data fragmentation techniques are aimed at enhancing the data manipulation process, decreasing processing time, facilitating data manipulation, optimizing storage, increasing flexibility, distributing processing costs, and facilitating data distribution and transportation, but are not specifically designed with data security in mind. Current state-of-the-art approaches that focus on security aspects in data fragmentation rely on encryption for ensuring data confidentiality. Our approach aims at minimizing the amount of encryption needed and relies on unlinkability of data entities to limit the privacy impact in case of single-point data leakages. Unlinkability is achieved by distributing parts of the original data to different storage providers. We focus on data fragmentation over relational databases which conform to the Normalization paradigm. The normalized data tables are regarded as standalone fragments that are then distributed to different Cloud storage providers. These storage providers are required to be non-colluding, which can be ensured by e.g. Service Level Agreements (SLAs) or legal regulations. Note that these regulations usually specify confidential quality requirements only, but no countermeasures. Although costs play a significant role in Cloud computing we do not refer to them as one of the prior goals of this paper. In order to be able to take additional confidentiality constraints depending on the data's domain into account, the data sets need to be analysed first, as proposed in , to ultimately create SLAs conforming to well-defined user-specific confidentiality requirements.

This paper is structured as follows: Section 2 discusses current confidentiality concerns and privacy issues for the outsourced data. Section 3 describes the proposed model for secure data outsourcing distinguished through fragmentation and distribution process. Section 4 depicts the application of our proposed model in a real world scenario. Section 5 concludes the paper.

II. BACKGROUND

Data storage is the central part of Cloud computing, which renders data confidentiality as one of the most critical issues in the Cloud. As Cloud customers entrust external service providers with their sensitive business data, from the customers' point of view, there is an obvious interest in keeping these data records protected within the shared environment of the Cloud storage. Thus, potential attacks in the Cloud need to be identified and analysed that could harm both the customers and providers. However, due to the differences in data domains, attacks and vulnerabilities are very context sensitive. Some Cloud services store customer information of very low sensitivity, while others host mission critical business functions or sensitive personal data such as health-related information. Any vulnerability of a particular service restricts user access to the data within the service until the provider can control and mitigate the vulnerability. In a survey,40.5% respondents stated that enterprises may seriously increase the risk of data leakage due to their focus towards using Software as a Service (SaaS) and Cloud computing solutions. As a consequence of data leakage,the customers' privacy can be seriously affected.

Privacy is the ability to decide when, how, and to what extent our information is shared. Data privacy needs to address the issues related to privacy protection goals including data subject rights, owner consent, data access, and anonymity. All these issues are considered according to the principles of Fair Information Practice . Privacy is a moral and legal right of each individual. Storing data and applications that reside at CSPs poses the potential risk of unauthorized access and processing of the data and application . Customers may lose control of their critical assets. Data confidentiality and privacy risks are even more critical when CSP reserve the right to change their terms and conditions .Unlinkability of the data has to be ensured when the data is distributed across different locations.

Data processing should involve only data that is actually required. As a consequence, the CSPs need to inform users about their data handling procedures and, in particular, what types of security measures are implemented to protect the data and how a data compromise is handled. In a stand-alone system, data confidentiality can be easily achieved, but as Cloud computing employs multiple databases to store multiple tenants' data, ensuring security and integrity-related paradigms such as transaction durability and consistency is very complex. It can be even more complicated when it comes to shared resources with multiple databases. Due to virtualization, privacy and security of transactions cannot be guaranteed by usage of the standard methods only, such as HTTP. Therefore, the Application Programming Interface (API) level is used for this purpose . However, APIs introduce additional complexities that can create security vulnerabilities in the API stack itself or in the technology responsible for handling the API calls. Therefore, secure channels and the reliability of the API-handling transactions are necessary for data integrity. Confidentiality also depends on how data is stored. The security of storage depends not only on the application responsible for storing the data but also on the operating system and other applications that run on the same system. The physical location of the stored data is a factor as well, as it is in the customers' interest to know where the data actually resides and whether the records are adequately secured. Data backups should be created regularly for fast recovery in case of a system or hardware failure. Finally, these backups need to be protected as well.

There are different types of threats related to data confidentiality such as inconsistent use of encryption and software keys, operational failures, authentication and authorization failures, or lack of security in data storage areas. Unauthorized deletions and modifications can severely impact customer data. These malicious actions may result in a multitude of consequences to both the customers and the service providers. While customers' privacy certainly takes a severe hit, service providers may suffer from loss of employee and partner morale and, of course, customer trust. Loss of core intellectual properties could also have competitive and financial implications.

In this paper, we focus on the customers', i.e., the data owners' view on data confidentiality and privacy. The overall goal is to protect outsourced data from both external attackers as well as from curious CSPs. The basic idea is to limit the expressiveness of the data entrusted to the storage providers by creating fragments to be distributed to different CSPs to ensure unlinkability. This enables the customers to limit the data exposure to semi-trusted, i.e., honest-but- curious service providers as long as these providers are non-colluding. This fragmentation and unlinkability approach reduces the need for potentially expensive and complex searchable encryption schemes , where the secret keys need to be maintained by the data owners themselves to ensure their privacy.

III. DATA FRAGMENTATION AND DISTRIBUTION MODEL

In this section we will discuss the technical and architectural details of our proposed model, including the definition and creation of the fragments according to privacy requirements and the distribution process.

A. Data Fragmentation model

Manipulating large amounts of data is a process that requires lots of computational resources. Facilitating efficient and fast data transport over the network is commonly ensured through one or several connections at the same time. We distinguish two types of data in relational database tables: Tables that define relationships and tables that contain actual payload data. The tables of the first type define relationships between payload data tables containing only foreign or primary keys linking payload tables. Regardless of the possibility that in some cases these tables can be connected to each other, they are stored on the same Cloud storage provider, since in case of an exposure, only an insufficient amount of data will be exposed. Table of the second type contain the actual payload data, i.e. the data that actually has to be stored and generates the fundamental structure of the relational database, connected with relationship tables. To decrease and optimize processing costs, data in relational databases is split up into smaller pieces, which we address as fragments . Our approach exploits fragmentation techniques for the purpose of ensuring data privacy and confidentiality, where fragmented data has to satisfy three essential requirements before being distributed to different storage areas:

1. The database has to be in third normal form before any kind of process is applied, so each table can be treated as independent fragment
2. Confidentiality levels define the importance of the data contained in a fragment
3. User Requirements provide a set of additional demands concerning the distribution of the fragments that can be chosen by the user

The first essential requirement, the database normalization process, is critical for ensuring independency and unlinkability of the tables. Normalization is used to control data redundancies and thereby reduces data anomalies. It is applied through several stages called normal forms: First Normal Form (1NF), Second Normal Form (2NF) and Third Normal Form (3NF). First normal form implies duplicate removal and creating separate tables for related data. Second normal form meets all requirements of the first normal form and fulfils separation of data subsets by placing them in separate tables and creating relational dependencies between them. Third normal form meets the first two normal form requirements, and additionally removes the columns that are not directly dependent on the primary keys. There are also higher level of normal forms used only in theory, but for our approach usage of the first three is sufficient. The objective is to ensure that each table conforms to the concept of well-formed relations that corroborate the following characteristics: i) each table can be an independent subject, ii) there will be no redundant data stored, iii) non-primary attributes depend only on the primary key, and iv) integrity and consistency of the data must be ensured.

In our approach we use a second essential requirement, the definition of the Confidentiality Level for each table, which is depending on the data that the respective tables contain. In our approach, we use three different confidentiality levels: High Confidentiality tables; Medium Confidentiality Tables and Low Confidentiality Tables. High Confidentiality tables (cf. Table 2) contain highly sensitive data, such as social security numbers, personal identification numbers or credit card numbers, which must be appropriately protected. In order to minimize the application of encryption, we consider the following approaches: We either store the data in the trusted local domain without encryption, or outsource the data while apply partial encryption(cf. Table 1). We consider the approach of applying the encryption on whole columns, even including the entity name, in order to hide the domain of stored data and to add some additional complexity regarding correlation attacks: For example, a user could store different types of data marked as highly confidential and encrypt them. In this case attacker cannot know if the data set holds credit card numbers or just some different, less sensitive, information. When using encryption, the encryption keys must be stored on local domain and be accessible to appropriate users only. The problem of the encryption approach is, that it increases computational cost, because each time data has to be decrypted before query processing.

Table 1: Encrypted data of highly confidential tables prepared for distribution

EmpID	536480aa02d0014998bba86ba20d5855
EN1-26	d78c3b972ad9cf53ddf8ac7144aea80c
EN1-35	c8e562a275af91a90accffd7180999c6
EN1-33	acb195087d0d5424c17724a425c07ba1
EN1-33	ddd97a3e5dbf2908fc1f58307e63f894
EN1-35	2d9abf9143f741b41dab9c9a38b6c318

Medium Confidentiality Tables, (cf. Table 3), contain tables that have to be treated carefully in case of dependencies with other tables and to distribute them accordingly. Low Confidentiality Tables,(cf. Table 5), are usually stored in a manner to provide efficient manipulation rather than to provide security because for this level, confidentiality is not a priority. Table 4 shows a fragment that represents a relational table that stores entity connections in a relational database. These Fragments have to be distributed to several servers, to preserve unlinkability.

The last essential requirement, User requirements, is a set of specially defined demands set by the user, that allow for some additional tailoring of the distribution process. The user may define additional detailed confidentiality demands for each table such as: Increasing or decreasing level of confidentiality, demanding partial encryption of designated columns, storing the fragments inside the local domain or even additional fragments splitting.

B. Data Distribution model

The data fragments as defined by the three essential requirements are distributed to different Cloud storage

Table 2: High Confidentiality table

EmpID	Credit Card Number
EN1-26	5500 6469 0000 0004
EN1-35	3000 9183 0000 0234
EN1-33	4111 1111 1111 1111
EN1-33	3400 6546 0000 0029
EN1-35	7010 9828 0000 0124

Table 3: Medium Confidentiality tables

ProjectNum	ProjectBudget
30-457-T3	25.896.500.000,00
31-124-T3	9.870.000.000,00
30-452-T3	10.000.000.000,00
30-482-TC	3.200.000,00
31-238-TC	150.000,00

Table 4: Example of relational table

ProjectNum	EmpID
30-452-T3	EN1-26
30-452-T3	EN1-35
30-328-TC	EN1-33
30-452-T3	EN1-33
31-238-TC	EN1-35

Table 5: Low Confidentiality tables

ZIP	City
10000	New York
11000	Washington, DC

11000	Washington, DC
14025	Boston
92000	California

providers as shown in Figure 1, to enforce data confidentiality in cloud regardless its locality and facilitates data distribution by using fragmentation concepts. The emphasis of this model lies on providing simple and secure data distribution to distinct Cloud storage providers that possess high computational and storage resources. In general, the distribution model distinguishes two domains, the trusted local domain where the data originates from and the semi-trusted public domain where the data are distributed to. The fact that the Local domain is considered trustworthy is used to solve the issue with tables that contain highly sensitive data without applying encryption. The public domain is considered non-confidential and thus the distributed data needs to be protected appropriately during the distribution process, both, at the distribution point and during use. To ensure secure and reliable transport between the customers local and public domains, we establish Virtual Private Network (VPN) sessions towards the insecure public domain. Insurance of appropriate levels of services and confidentiality from the Cloud storage service providers is established by using SLAs. The SLAs encapsulate all three essential requirements along with performance, availability and serviceability requirements, which the user demands for desirable and efficient data outsourcing.

The local domain constitutes the starting point of the data distribution process. The distribution process is illustrated with the fragmentation algorithm stated below. The first step of fragmentation process deploys each table (t) of a database schema S, as an individual fragment, (step one of the Fragmentation algorithm). To ensure efficient transport and compatibility with different types of relational databases, the data is exported to XML files. Depending on the use case scenario, the user defines a set of essential user requirements (data availability, serviceability, performance), which depend on the data that the fragments contain, (step two of the Fragmentation algorithm).

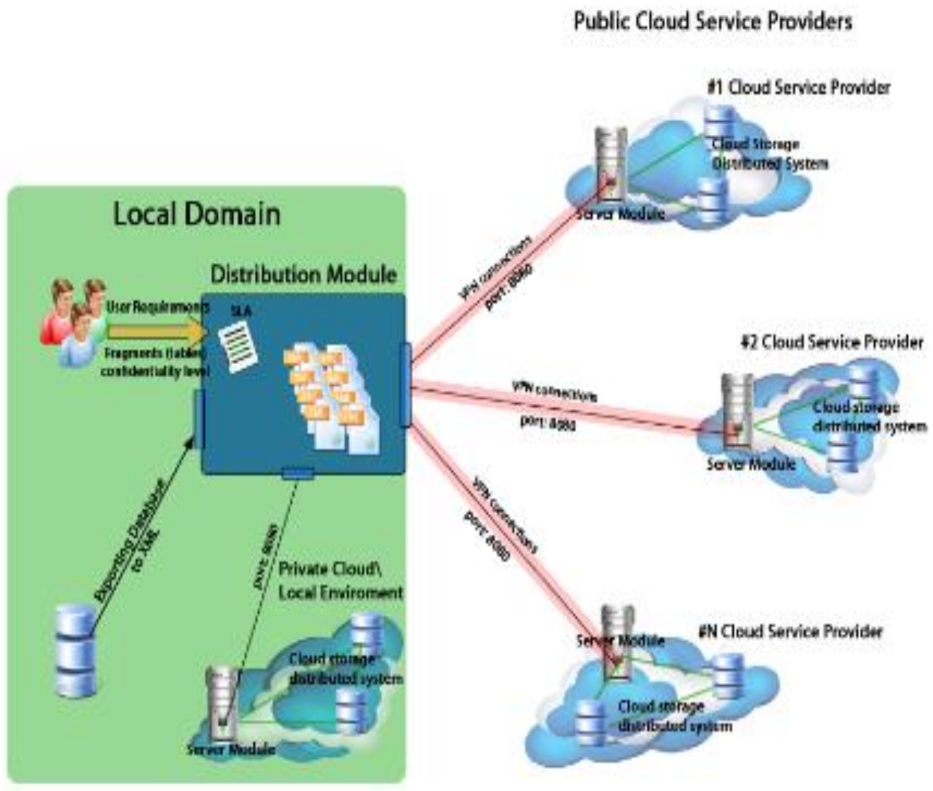
Public Cloud Service Providers
#1 Cloud Service Provider
Cloud Storage Distributed System
Server Module
Local Domain
Distribution Module
User Requirements
SLA
Fragments (tables) confidentiality level
VPN connections
port: 8080
#2 Cloud Service Provider
VPN connections
port: 8080
Cloud storage distributed system
Server Module
VPN connections
port: 8080
Exporting Database to XML
port: 8080
Private Cloud\ Local Enviroment
Cloud storage distributed system
Server Module
Server Module
#N Cloud Service Provider
Cloud storage distributed system

Figure 1: Architecture of the Proposed Model

Afterwards, requirements are assigned, along with confidentiality levels, for each fragment with respect to the data that they contain, (step three of the Fragmentation algorithm).

1. Fragmentation process: In this step, the tables in 3NF are exported into fragments, e.g. in XML-files. For the rest of the algorithm, we let T = {ti , i = 1, . . . n} be the set of all fragments.

```
for t ∈ Tn do
for j := 1 to l step 1 do
if ((t ∩ ts = ∅, ∀ts ∈ Pj : CL(ts ) > 1) ∧ ((t, ts ) ∉ U, ∀ts ∈ Pj ))
Pj := Pj ∪ t
break
```

```
end
for i := 1 to n do
    Deploy fragment ti as independent fragment.
end
end
```

2. Requirements definition: Defining a set of user-defined additional distribution requirements. In this step the user can decide not to distribute certain fragments together. The user requirements are given in the form (t_i , t_j), where i = j, denoting that the fragments t_i and t_j should not be distributed together. Let $U_0 = \{(t_i , t_j)\}$ be the set of all user- set requirements, then we define U to be the set of all user requirements with $U = U_0 \cup \{(t, t_0) \mid (t_0 , t) \in U_0 \}$.

3. Independent fragment deployment - Fragmentation process:

```
for i:=1 to n do
    Assign confidentiality level to fragment ti , i.e. set CL(ti ).
end
```

4. Selection of CSPs: In this step the estimated number l of CSPs required for the distribution of the fragments is calculated. Additionally, the l CSPs $P_1 , . . . , P_l$ are selected.

5. Assigning Fragments to CSPs: The following algorithm targets at constructing an assignment that distributes the fragments with respect to the user requirements. Additionally, tables that share columns and where at least one of them possesses a classification level higher than 1 are not allowed to be distributed to the same CSP.

```
P := (P1 , . . . , Pl )
Tc := {t ∈ T |CF(t) > 1}
Tn := T \ Tc
for t ∈ Tc do
    for j := 1 to l step 1 do
        if (((t ∩ ts = ∅) ∧ ((t, ts ) ∉ U )), ∀ts ∈ Pj )
            Pj := Pj ∪ t
End
```

6. Distribution process: The following algorithm distributes the fragments to the appropriate CSPs. In this step, the encryption for fragments with classification level 3 is applied.

```
for Pi ∈ P do
    for t ∈ Pi do
        if CL(t) = 3 then encrypt(t) → ProviderPi
        else t → ProviderPi
        fi
end
```

The next step, (step four of the Fragmentation algorithm), calculates an upper bound for the number of Cloud Storage providers required for distributing the data, by exploiting the relationship between primary and foreign keys. The primary key is used to uniquely define each row and it is also defined to establish the relationship with other tables, where it is defined as a foreign key (FK). The Foreign Key is used to reference connectivity of one table to another by mapping itself to the appropriate primary key:

$$N_s = \sum_{i=1}^{n} \mathrm{FK}_i + 1,\ \mathrm{FK}_i \in \{0,1\}, i \in \{0,1,2,3,...n\}\ (1)$$

where the FK_i represent foreign keys, N_s represents the number of Cloud storage providers in the worst case scenario and n number of FK_i mapped on PK_i .

The formula shown in(1) determines the number of foreign keys that map themselves to primary keys, and therefore, also determines directly related tables. To complete the count, we add one more unit that represents the table containing the primary key. The table that has the largest number of mappings (N_s) gives us the required number of Storage service providers in the worse case scenario. Now, when we have the number of CSPs for the worst case scenario, we use user requirements and confidentiality levels of tables in order to decrease the number of CSPs actually needed. Because the number of CSPs was calculated by using the table with the most primary key - foreign key connections, we evaluate that table along with the connected ones to decrease the number of CSPs if possible. Each set of tables that can not be correlated directly by primary or foreign keys can be distributed to the same CSP, so it automatically means that we can have less CSPs then calculated directly by the formula. After evaluation of each correlated table, we determine the minimum number of CSPs that necessary for the distribution.

The Fragmentation process, which is also illustrated in the Distribution model, Figure 2, is applied for each fragment (table) regarding confidentiality level, User requirements and ER model.

The ER model helps to better illustrate the relational point of view between tables and to facilitate the fragmentation process. Each table is analysed and designated to corresponding distribution groups,(step five of the Fragmentation algorithm). Each distribution group represents one CSP, where the data is going to be distributed. The distribution process, (step six of the Fragmentation algorithm), provides secure and confidential data transport by usage of VPN connections between the customer and provider.

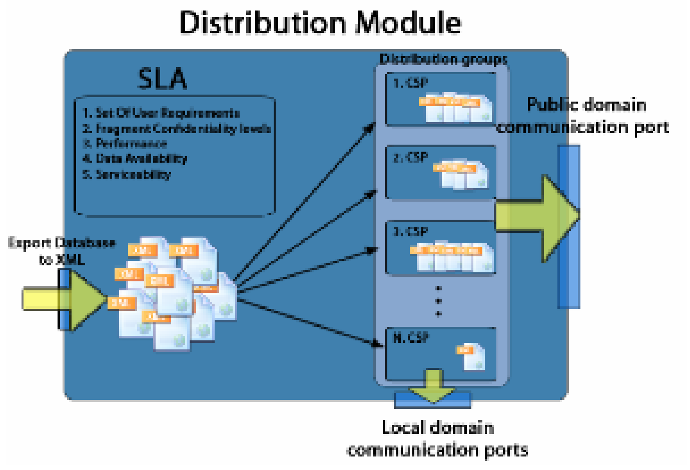

Figure 2: Distribution module

IV. CONCLUSION

Cloud computing offers attractive and cost effective solutions to customers, but it also imposes many challenges regarding confidentiality, privacy, control and laws and legislation. Most of the security measures are based on trust reliance where the customer relies strongly on the providers' trustworthiness. This paper focuses on secure and confidential data outsourcing to Cloud environments by using fragmentation techniques and applying only minimal encryption to prevent data exposure. It relies on database normalization, user requirements and confidentiality levels in order to enforce privacy before distribution. Exploitation of the database normalization process ensures an appropriate fragment structure without any redundancies and anomalies, by assuming that relational tables are in 3NF. We base our approach on the relational databases, as the most common database paradigm of data persistence. Each table is analysed to determine data sensitivity and designate appropriate confidentiality levels, as the basis for the fragmentation process to exploit tables as independent fragments. The appropriate number of CSPs are determined, after which we designated the fragments according to user requirements and confidentiality level. To distinguish the real contribution of the Cloud's features, to make a decent thorough evaluation and to perform simulation of performance (e.g., availability, overhead costs, scalability, etc.) , we will run experiments over a longer period of time. Because of the flexibility of Cloud providers it is relatively easy to make tests concerning data distribution, but to run the complex test that will include Scalability of a few parallel systems (CSPs) along with data distribution between them, takes lot effort and period of time. Thus we have reserved this task for future work were we will also discuss the generation of a decent test environment and network limitations.

References

[1] Amazon, Amazon elastic compute cloud (amazon ec2), http://aws.amazon.com/ec2/.

[2] Huang Bin and Peng Yuxing, A novel metadata management scheme in cloud computing, International Conference on Software Technology and Engineering(ICSTE) (2010), V1–433 – V1–438.

[3] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic, Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (2009), 599–616.

[4] Yanpei Chen, Vern Paxson, and Randy H. Katz, Whats new about cloud computing security?, Tech. Report UCB/EECS-2010-5, EECS Department, University of California, Berkeley, Jan 2010.

[5] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, Controlling data in the cloud: outsourcing computation without outsourcing control, Proceedings of the 2009 ACM workshop on Cloud computing security, ACM, 2009, pp. 85–90.

[6] Valentina Ciriani, Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati, Combining fragmentation and encryption to protect privacy in data storage, ACM Trans. Inf. Syst. Secur. 13 (2010), 22:1–22:33.

[7] Google, Google app engine, http://code.google.com/appengine/.

[8] Albert Greenberg, James Hamilton, David A. Maltz, and Parveen Patel, The cost of a cloud: research problems in data center networks, SIGCOMM Comput. Commun. Rev. 39 (2008), 68–73.

[9] Bernd Grobauer, Tobias Walloschek, and Elmar Stocker, Understanding cloud computing vulnerabilities, IEEE Security and Privacy 9 (2011), 50–57.

[10] IBM, Ibm smart cloud, http://www.ibm.com/cloud- computing/us/en/.

[11] Lori M. Kaufman, Data security in the world of cloud computing, IEEE Security and Privacy 7 (2009), 61–64.

[12] Sean Carlin Kevin Curran and Mervyn Adams, Security issues in cloud computing, Elixir 38 (2011), 4069– 4072.

Authors

First Author: A.Sabina Parveen, Syed Ammal Engineering college, Ramnathapuram, Tamil Nadu, India
Email id - **Sabina_2k@yahoo.com**

Second Author: C.R.Suganya, Syed Ammal Engineering college, Ramnathapuram, Tamil Nadu, India