

A Metadata Based Storage Model for Securing Data in Cloud Environment

Subashini, S. and V. Kavitha
Department of Computer Science and Engineering,
Anna University Tirunelveli, Tamilnadu, 627007, India

Abstract: Problem statement: Enterprises are migrating to the cloud environment at a faster pace. Security of information that is being processed by the applications and ultimately getting stored in the data centers are of big concerns of this newly evolving environment. The security of the data is a concern not only during transferring of data through the wires but also during its storage phase where data stays most of the time. **Approach:** In order to keep the data secure during its storage phase, a preventive, robust security model is required. Instead of developing a robust security module to prevent hackers from intruding into data centers, a model which will prevent intruders from getting the required information even at the event of intrusion, will be of utmost use. Conventional security models secure data by encryption or by fragmentation. A security model developed using a fragmentation technique that is based on the sensitivity, criticality and value of the data provides better security by means of disintegration of value of the data and also a good technique for prevention of information leaks. The proposed method also provides solutions to access the fragmented data. **Results:** The proposed model provides an efficient security solution for data stored in cloud. When compared to conventional methods, the speed of data queries are less for small databases, but prove to be very efficient for huge databases. **Conclusion:** This model provides an efficient solution for data storage security in cloud environment. This technique coupled with standard encryption techniques will make this model more robust.

Key words: Cloud computing, data privacy, data protection, data storage, data security

INTRODUCTION

Even as an increasing number of firms look at embracing cloud computing, the security of data predominantly remains as a primary concern. Cloud requires security which depends and varies with respect to the deployment model that is used, the way by which it is delivered and the character it exhibits. Some of the fundamental security challenges are data storage security, data transmission security, application security and security related to third party resources (Subashini and Kavitha, 2011). As this new generation infrastructure gains momentum, more and more applications and data are moved to this untested environment. Though the underlying infrastructure of the system paves way for elasticity and easy deployment of the services by vendors, this mounting opportunity has a trailing risk which poses a major risk and concern over the system's security. Cloud computing moves the application software and databases to the large data centers, where the management of the data and services are not trustworthy. This unique attribute, however, poses

many new security challenges (Wang *et al.*, 2009). These security concerns should be curtailed at its root instead of deploying much effort at the later stages when the system is scaled beyond imagination and solutions are outside implementable limits. To realize this tremendous potential, business must address the privacy questions raised by this new computing model (BNA, 2009).

This study proposes a methodology for securing data that is being stored at data centers and other locations of the cloud. The data under consideration is inclusive of data that is residing in a database and as well as in the file system. The life time of the data at the storage location is obviously more than the time it is over transmission. Though data transmission security is of importance, the security of the data at the stored location is of utmost importance. Hence we propose a methodology to secure data during its time it is being residing in the storage location. This inherently triggers the need for designing ways to store and retrieve data. The rest of the study unfolds this methodology and is organized as follows: In this study firstly we discuss about related works in this area. Next this study

Corresponding Author: Subashini S., Department of Computer Science and Engineering, Anna University Tirunelveli, Tamilnadu, 627007, India

describes the overall functionality of this methodology. Later we explain ways to design data storage and access methods to maintain integrity of data. Finally we provide a sample implementation algorithm and also different elements that should be considered for storing data in database. It also lists out generic concerns that have to be taken into account to adhere to the proposed model.

Related works: Ignoring fragmentation with respect to providing security, data fragmentation is not a new concept. Concepts like these are already in use for providing optimization of data access in distributed systems. But most of them do not take security as the concern for fragmentation. One such work is regarding fragmentation and allocation of data in distributed database systems (Hose and Schenkel, 2010). Here they propose a model to fragment data horizontally or vertically with relation to the tuples so that data can be accessed or updated in an optimized manner.

Another work proposed by Fabre and Perennou gives a model based on object fragmentation at design time to reduce processing in confidential objects. They give an idea that the more non confidential objects can be produced at design time, the more application objects can be processed on untrusted shared computers (Fabre and Perennou, 1995). In another study Gibbs describes about different problems created by the fragmentation of information across a number of different databases that are maintained and controlled by different function units within an organization (Gibbs *et al.*, 2005).

These algorithms provide optimal ways to re-arrange and access data that are fragmented and stored in different locations. The main concerns in these works are to fragment data on the basis of easy retrieval but not relating to providing security to the data under consideration. Fragmentation of data based on relevance to data value is not targeted in any of the works. Fragmentation based on meta data is used in some works but those considerations are truly based on relevance to optimize data access rather than to the security of the data itself.

Metadata based data storage model: This model is based on the fact that any information is valuable only as long as the fragments of the information are related to each other. When related information are not available in a mapped manner, it is of no use. For example, information about a credit card number without its corresponding information like card holder name, validity date information and Card Verification

Value (CVV) is invaluable and so is it's vice versa. And a similar example is the mapping of username and password. A username alone is not valuable and so is the information about the password alone. The information becomes valuable only when these fragments of information are mapped. The mapped information about elements is required only for authenticated users and owners of the respective information. A well known instance of intrusion of user information is the one recorded by Sony PS Network in recent times (Goodin, 2011).

In such a scenario, there is no necessity that data should be stored in a mapped manner. But mapping is needed at the point of usage. Juels and Kaliski described a formal "Proof of Retrievability" (POR) model for ensuring the remote data integrity. Their scheme combines spot-checking and error-correcting code to ensure both possession and retrievability of files on archive service systems (Juels and Kaliski, 2007). The time of usage of the information is apparently very less in comparison to the time that data is present at the storage location. Thus two types of security concerns arise. One concern is during data usage, i.e. during transmission and secondly, static phase of the data, i.e. during residing at storage centers. With respect to the data security during transmission in the cloud we have proposed a layered framework to deliver security as a service in cloud environment (Subashini and Kavitha, 2011). This framework consists of a security service which provides a multi-tier security based on the need of the transaction. The framework provides dynamic security to users based on their security requirements, thus enabling localized level of security and thereby reducing the cost of security for applications requiring less security and providing robust security to applications really in need of them.

The model described in this study only deals with the data security at the storage centers. This in turn has two concerns: One issue is about the actual physical unit where the data is stored and the other one is the intrusion into the information. Our model is mainly focused in providing security in avoiding intrusion. This model does not prevent hackers from getting hold of the data. Rather it makes the data invaluable even if it is accessed by an intruder.

To adhere to this model, care has to be taken right from the design phase of the information storage. Data has to be segregated into Public Data Segment (PDS) and Sensitive Data Segment (SDS). The SDS has to be further fragmented into smaller units until each fragment does not have any value individually.

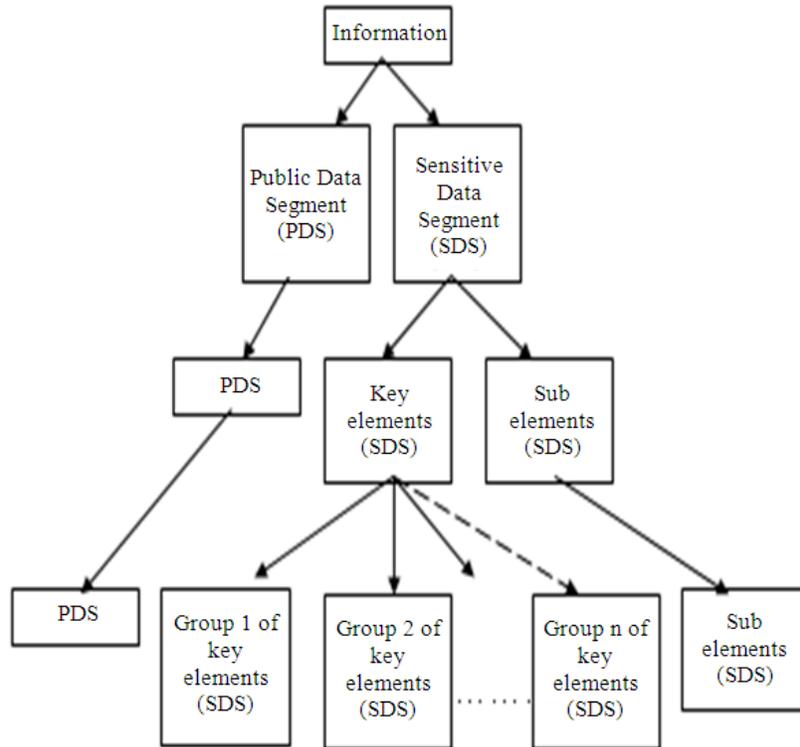


Fig. 1: Data fragmentation

The fragmentation need not be of multiple levels. Instead, effort has to be put in to identify the key element that makes the data sensitive and should be fragmented separately. Figure 1, explains this fragmentation.

The value of the information is actually destroyed in this process. But as and when fragmentation is done, the mapping data required to re-assemble the information should also be generated parallelly. This can be done for database that is being designed from scratch. But, this is not effective for enterprises who want to move their existing data to the cloud. As a measure of migration of data from existing environment to cloud, the migration should be done appropriately. This can be made feasible by this model. For achieving this, we need a Data Migration Environment (DME) which does this job. The input to DME should be the existing schema of the database and additionally information about the sensitive part of the schema should be given as Metadata to the DME. The DME can fragment the data into pieces based on the level of security needed. Along side it will prepare a mapping table to re-assemble the data. The functionality of this environment and considerations for data integrity are discussed next..

The methodology: Let us consider our previous example of credit card information and roll out our methodology using this example.

Consider a database in a bank consisting of user information along side with the credit card information. The schema for storing such information will be in the form of tables with some tables containing personal information of the user and some tables containing information regarding to credit cards and will be mapped using their ids.

This particular information can be stored in a database (say bankDb) this way:

bankDB:

- A Customer table containing
 - CustomerId (Primary Key (PK)),
 - CustomerName,
 - CustomerAddress,
 - CustomerPhone,
 - CustomerDOB
- A Membership table containing
 - CustomerId (Primary & Foreign Key (FK))
 - Password,
 - PasswordQuestion,
 - PasswordAnswer

- A Creditcard table containing
 - CardId, (Primary Key)
 - CreditcardNo,
 - CardExpiryDate,
 - CVVNo
- A Customer_Creditcard table containing
 - CustomerId (Primary Key)
 - CardId (Primary Key)

An intruder who gets access to this particular database can exploit this information because all related information are stored at the same location.

In this example, the Customer table contains data which is not of much importance. The Membership table taken individually does not have any value but along with the Customer table data, it is juicy information for an intruder. The Creditcard table is a sensitive data with high value because though there is no mapping done with the Customer table, it individually is a high potential target. For example, an online transaction can be done successfully with this data alone. And together with the information of Customer table and Customer_Creditcard table, the bank can become bankrupt overnight. Usually, the entire data is stored in a single database and most probably on the same hardware resource.

Our model enforces that the related data should be stored at different locations and should be mapped runtime either during update or query. Consider that this entire model is migrated to our proposed model through the DME. The user has to supply the schema information of these tables to the DME and along side its metadata. Let us consider only three categories of metadata for this example. The data which is having low value is considered as 'Normal'. The data which is having high value is considered as 'Critical' and the data which has value when mapped with other data is considered as sensitive. And the data which maps 'Sensitive' or 'Critical' data to 'Normal' data is also considered 'Sensitive'. The metadata for our example are shown in Table 1.

The DME now has to fragment this data. The DME should be able to be configured or customized with respect to the level of security required. Considering our example, if we want the DME to provide medium level security, it should fragment only data which are of 'Critical' criteria. And if high level security is required, it should fragment data present in both 'Critical' and 'Sensitive' criteria. The DME is not aware of the actual data residing within these tables. Hence along with the metadata of the tables, the primary key column name should be provided in addition to it.

Table 1: Metadata information

Table	Metadata
Customer	Normal
Membership	Sensitive
Creditcard	Critical
Customer_Creditcard	Sensitive

Table 2: Metadata information after fragmentation

Table	Metadata
Customer	Normal
Membership	Sensitive
DME_Creditcard	Sensitive_DME
Customer_Creditcard	Sensitive
DME_Creditcard_Sensitive	Sensitive_DME
DME_Creditcard_Mapper	Sensitive_DME

Table 3: Segregated schema

Normal	Originally sensitive	Sensitive DME
Customer	Membership	DME_Creditcard
	Customer_Creditcard	DME_Creditcard_Sensitive
		DME_Creditcard_Mapper

This is easily available with the schema information of the database tables. The different levels of security needed and their corresponding metadata should be configured with the DME.

Let us consider that we need medium security for our database. Then the DME can fragment only the data that is 'Critical'. In our example, we have one 'Critical' data set. The corresponding table is Creditcard table and the primary key of this table is CreditcardId. As a first step the DME fragments this table as below:

- DME_Creditcard table
 - SensitiveId (PK, Created by DME)
 - CreditcardNo
 - CardExpiryDate
- DME_Creditcard_Sensitive table (Created by DME)
 - SensitiveId (PK, FK, Created by DME)
 - CVVNo
- DME_Creditcard_Mapper table (Created by DME)
 - CreditcardId (PK)
 - SensitiveId (PK, Created by DME)

Now when we look into the data of the above three tables all of them will fall under the 'Sensitive' category of metadata. Table 2 lists the metadata of the database at this current situation.

After fragmentation is completed, the DME segregates the schema, separating out the data modified by DME, 'Originally Sensitive' data and 'Normal' data as shown in Table 3.

Table 4: DME_MAPPER Table

OriginalTableName	NewTableName
Creditcard	DME_Creditcard
Creditcard	DME_Creditcard_Sensitive
Creditcard	DME_Creditcard_Mapper

Then the 'Sensitive DME' data is then split into Actual Data (AD) and Mapper Data (MD):

- Sensitive DME
 - Actual Data
 - DME_Creditcard
 - DME_Creditcard_Sensitive
 - Mapper Data
 - DME_Creditcard_Mapper

The DME then moves the 'Normal' data to one database and 'Originally Sensitive' data to another database and AD of 'Sensitive DME' data to another database at different location and MD of 'Sensitive DME' to the database with 'Normal' data. With respect to the AD, if DME creates its own table, then this table will be the most sensitive data and will be stored in a different location. Different location here means either different server at the same geographical location or at different geographical location. Additionally one more mapping is required for mapping the original table with the fragmented data set. This can be stored in a separate table. Now the database looks like the following:

Server 1

bankDB:

- Customer table containing
 - CustomerId (Primary Key (PK)),
 - CustomerName,
 - CustomerAddress,
 - CustomerPhone,
 - CustomerDOB

bankDB_DME

- Membership table containing
 - CustomerId (Primary and Foreign Key(FK))
 - Password,
 - PasswordQuestion,
 - PasswordAnswer
- Customer_Creditcard table containing
 - CustomerId (Primary Key)
 - CardId (Primary Key)
- DME_Creditcard_Mapper table containing
 - CreditcardId (PK)
 - SensitiveId (PK, Created by DME)
- DME_Mapper table containing
 - OriginalTableName (Combined PK)
 - NewTable Name (Combined PK)

Server 2

- DME_Creditcard table
 - SensitiveId (PK, Created by DME)
 - CreditcardNo
 - CardExpiryDate

Server 3

- DME_Creditcard_Sensitive table (Created by DME)
 - SensitiveId (PK, FK, Created by DME)
 - CVVNo

The DME_Mapper table is shown in Table 4.

Now each database contains data which does not have value in itself. The entire mapping is done only during runtime and the value is built up temporarily during access and update and later its value is destroyed. An intruder who gets access to the data during the static phase of the life cycle of the data can not use the data to exploit the information by any way. The integrity between the original schema and the new schema can be taken care by deploying a database runtime migration environment which will deploy all the logics required for the runtime generation of schema and its corresponding drop after its lifecycle.

Implementation and cost: A typical algorithm that will be used for fragmentation is as follows:

Assuming No. of Tables as 'n' and No. of Data Servers (DS) as 's'

For k = 1 to s

DS[k].used = false;

End For

For i=1 to n

getMetaDataSensitivity(Table[i])

End For

For i=1 to n

If(Table(i).Sensitivity == Normal){

DS ds = getUnusedDS()

StoreTableInDS(ds,Table[i])

'StoreTableInDS also stores the information of the tables 'stored in the DS in a hashtable which will be used by the 'runtime environment to re-create the table dynamically 'during runtime access

continue;

}

else if(Table(i).Sensitivity==Sensitive){

If(requiredSecurity==High){

DME_Table[] dme_t_high = Split(Table[i])

DME_MapperTable dme_map_t = _

CreatedMEMapperTable(Table[i],dme_t_high

)

DS ds = getUnusedDS()

StoreTableInDS(ds,dme_t_high)

```

        ds = getUnusedDS()
        StoreTableInDS(ds,dme_map_t)
    }
}
else if(Table(i).Sensitivity==Critical){
    DME_Table[] dme_t = Split(Table[i])
    If(requiredSecurity==High){
        SplitFine(dme_t)
    }
    DME_MapperTable dme_map_t = _
    CreateDMEMapperTable(Table[i],dme_t)
    DS ds = getUnusedDS()
    StoreTableInDS(ds,dme_map_t)
    'Assuming No. of DME Tables as 'm'
    DS ds_sensitive = getUnusedDS()
    DS ds_critical = getUnusedDS()

    For j=1 to m
    If(dme_t[m].isDMESensitive == false){
    StoreTableInDS(ds_sensitive,dme_t[m])
    else
    StoreTableInDS(ds_critical,dme_t[m])
    }
    End For
}
End For

```

During querying of data, the runtime environment uses the hash table containing information of the fragmented tables to restructure the input query by replacing and inserting a join query with the input query and then executing it to form tables with original relationships of the data and once the dynamically created tables are destroyed after the access is over. The fragmentation of data incurs a cost overhead which can be calculated as follows:

- C1 = Cost of Fragmentation of one Critical table * No of Critical tables
- C2 = Cost of Fragmentation of one Sensitive table * No of Sensitive tables (this cost incurs only when required security is high)
- C3 = Cost of Creating a Mapper Table * (No of Critical tables + No. of Sensitive Tables)
- C4 = Cost of regeneration of fragmented tables * ([No of Critical table*No of DME tables created newly] + [No. of Sensitive table*No of DME tables created newly])
- C5 = Cost of encrypting/decrypting total database
- C6 = Cost of fragmenting data for data dispersal in distributed systems

Total Cost of Security without fragmentation T1 = C5 + C6.

Where:

$$C6 \leq C1+C2+C3+C4$$

C7 = Cost of encrypting / decrypting Sensitive Tables.

Where:

$$C5 > C7$$

Total cost of Security with fragmentation and encryption when compared to security using only encryption and fragmentation for data dispersal purpose $T = C1+C2+C3+C4+C7-T1$ where $T > T1$.

The cost of this method is more than traditional methods but it provides a better security. Since this model will be deployed in a cloud which is conceptually an environment with high processing power, the cost incurred will provide proper justification when compared to the security it provides. Data dispersal and Data fragmentation are some of the techniques that can be attempted with ease with the cloud environment.

MATERIALS AND METHODS

The entire technique can be reproduced both in a simulated environment and real time environment. The DME can be implemented in any of the programming languages. This DME implementation can be segregated into migration environment and data schema access layer. The migration environment should enable the user to define the metadata of the schema and data schema access layer should read the metadata of the schema and should do the fragmentation. The database schema can be designed either in Oracle or SQL Server. The pseudo code described in this study can be used for implementing the logic for fragmentation of data based on meta data of the schema. The data storage should be done in a distributed database and the mapper tables should be encrypted and stored in a distributed environment. The ultimate aim of the fragmentation is that the data which can be coupled to form a significant value should not be stored together.

RESULTS

A concrete implementation was made to test this methodology. A typical financial institution was taken into consideration and its database schema was designed. There were 17 Master tables and 41 Transaction tables. These tables were initially designed using normal methodology and then the schema was redesigned based on the model described in the study. A simple data migration environment was implemented as an application and the redesign of the database was done using this environment. There were 34 critical entities, 79 sensitive entities and the remaining were normal entities.

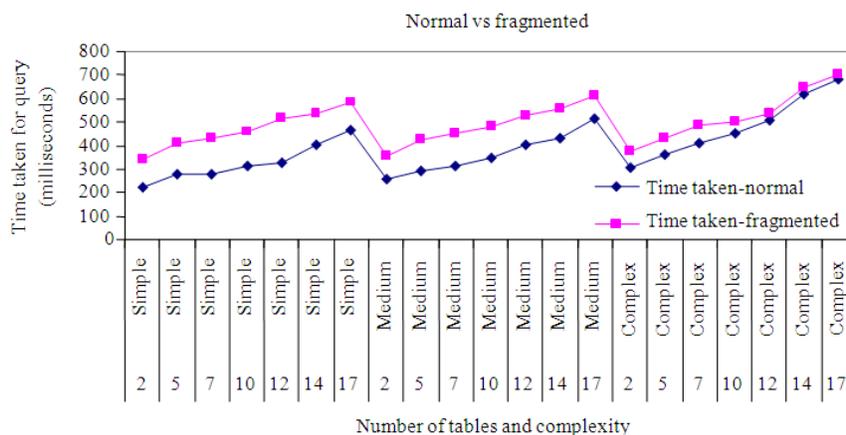


Fig 2: Performance between normal and fragmented environment

The DME created 22 new subtables to fragment the sensitive and critical data and in the process created 9 mapper tables. The cloud environment was simulated using Eucalyptus and 5 Sql Server databases were deployed in different machines in the cloud environment which totally contained 8 machines running on Ubuntu.

The resultant data from the DME is a set of queries that has to be run on the 5 databases and also a set of stored procedures which translates the original queries to queries required to form data from the fragmented data. The DME suggests the minimum number of distributed databases that is required for storing the fragmented data based on the original database schema. For our scenario, the DME suggested 3 databases. But the DME was asked to generated distribution based on 5 databases and hence the more number of subtables. The scripts generated by the DME was subsequently run on the database and the required database schema was created in those databases.

With this setup, the performance of the database and the integrity of the queries were tested. For this requirement, the normal database schema without fragmentation was setup in a separate database. Initially, only simple queries were made from both the environments (like queries involving data from 2 or 3 master tables). The time difference between the two environments were significant with the fragmented environment consuming more time and it was expected. Then as the complexity of the queries increased, the time difference became less significant and this was mostly attributed to the parralel querying of the sql server in a single machine in the normal environment with the parallel querying in multiple sql servers residing in different machines in the fragmented environment. The graph in Fig. 2 explains the time

taken for different type of queries in the normal and fragmented environment.

DISCUSSION

The limitations in this model are the initial effort taken to configure the DME and then migration of the existing data to the new model. Changes to the existing conventional database engines are unavoidable, because there will be an inherent need for plugging in the DME and the database runtime migration environment to these engines. There is a cost which is incured due to fragmentation of data. This cost includes cost of fragmentation of data while storage and also cost of forming the data at runtime from the fragmented data. But this cost is not newly introduced to the system because data fragmentation is already a practical methodology that is followed for distributed systems. Here the fragmentation is provided to make the data secure. In addition to fragmentation, a proper encryption technique can be used to provide additional security. This encryption can be done only to data that is fragmented as 'sensitive' by the DME. This reduces the cost of encryption of the entire database.

CONCLUSION

In this study we investigated the issues in security in data storage in cloud environment. To ensure that the data is secure during the stored phase of the life cycle of the data, we proposed a metadata based model using which the data residing at data center are robbed of their values and the values are temporarily built up during runtime and then destroyed once its usage scope is completed. This makes the data invaluable even if an intruder gets access to this data. Though this model will

take some quantifiable effort to be implemented in real time, it provides necessary solution for an environment like the Cloud which is showing an adverse potential to become the next generation enterprise environment. Implementing such a model during the earlier phases of the evolution of the system will be relatively easier with respect to implementing it after lot of data take refuge in the cloud. This model in combination with our multi-tier security model for securing data over transmission will provide proper cross bars in the wires of malicious users.

REFERENCES

- Goodin, D., 2011. User data stolen in Sony PlayStation Network hack attack. *Ars Technica*
- Fabre, J.C. and T. Perennou, 1995. Fragmentation of confidential objects for data processing security in distributed systems. *Proceedings of the 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, Aug. 28-30, IEEE Xplore Press, Cheju Island, pp: 395-403. DOI: 10.1109/FTDCS.1995.525009
- Gibbs, M.R., G. Shanks and R. Lederman, 2005. Data quality, database fragmentation and information privacy. *Surveillance Soc.*, 3: 45-58.
- Hose, K. and R. Schenkel, 2010. *Distributed Database Systems Fragmentation and Allocation*. Distributed Database Systems.
- Juels, A. and B.S. Kaliski Jr., 2007. Pors: Proofs of retrievability for large files. *Proceedings of the 14th ACM Conference on Computer and Communications Security*, Oct. 28-31, ACM Press, USA, pp: 584-597. DOI: 10.1145/1315245.1315317
- Subashini, S. and V. Kavitha, 2011. A survey on security issues in service delivery models of cloud computing. *J. Netw. Comput. Appl.*, 34: 1-11. DOI: 10.1016/j.jnca.2010.07.006
- Wang, C., Q. Wang, K. Ren and W. Lou, 2009. Ensuring data storage security in cloud computing. *Proceedings of the 17th International Workshop on Quality of Service*, Jul. 13-15, IEEE Xplore Press, Charleston, SC., pp: 1-9. DOI: 10.1109/IWQoS.2009.5201385
- BNA, 2009. *Privacy and Security Law Report*. The Bureau of National Affairs, Inc.