Journal of Engineering and Applied Sciences 7 (8-12): 483-489, 2012
ISSN: 1816-949X

A Metadata Based Storage Model for Securing Data in Cloud Environment

S. Subashini and V. Kavitha
Anna University, Tirunelveli, Tamil Nadu, India

Abstract: IT 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. And the architecture that is needed to secure the stored data is of much importance than while the data is getting transferred because of the fact that the data resides relatively for a long time in the storage area than in the wires. To ensure the security of the data stored in the data centers, researchers propose a new methodology which might not completely help in restricting a hacker to access the data but will make the data invaluable if it is extracted by a hacker but at the same time ensures the quality of the data that is being provided to its respective owner or authorized user. Researchers propose a metadata based data segregation and storage methodology and also solutions to access this segregated data. This methodology ensures that data is invaluable during static residence and gains value only during acquisition or updation. This methodology was implemented and tested and its results are presented.

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

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 systems 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, researchers 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.

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 study 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 research is related to enhancement of ADRW algorithm to achieve dynamic fragmentation and object allocation in distributed databases (Sleit *et al*., 2007). Here, they deal more about the cost involved in accessing data fragments from remote sites.

Corresponding Author: S. Subashini, Anna University, Tirunelveli, Tamil Nadu, India

Another research proposed by Fabre and Perennou (1995) 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 research by Gibbs *et al*. (2005) 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.

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 study. Fragmentation based on meta data is used in some study 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 its 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 (Fabre and Perennou, 1995).

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 (Gibbs *et al*., 2005) 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. 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 researchers have proposed a layered framework to deliver security as a service in cloud environment. 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. The 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. 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 shows this fragmentation.

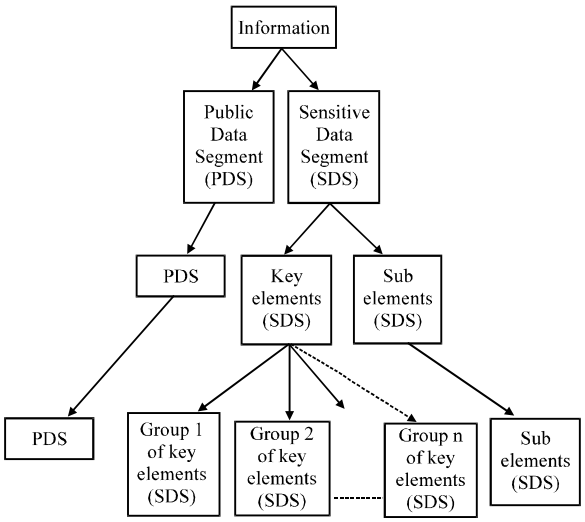

Fig. 1: Data fragmentation

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 parallely. 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 in this study.

THE METHODOLOGY

Let us consider the previous example of credit card information and roll out the 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))
- Customer name
- Customer address
- Customer phone
- Customer DOB

A membership table containing:

- CustomerId (primary and Foreign Key (FK))
- Password
- Password question
- Password answer

A creditcard table containing:

- CardId (primary key)
- Credit card No.
- Card expiry date
- CVV No.

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.

The 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 3 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 the 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 the 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

Table 1: Metadata information

Table	Metadata
Customer	Normal
Membership	Sensitive
Creditcard	Critical
Customer_creditcard	Sensitive

in addition to it. 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 researchers need medium security for the database. Then, the DME can fragment only the data that is critical. In the example researchers 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.

DME_creditcard table:
- SensitiveId (PK, created by DME)
- Creditcard No.
- Card expiry date

DME_creditcard_senstive table (created by DME):
- SensitiveId (PK, FK created by DME)
- CVV No.

DME Creditcard Mapper table (Created by DME):
- CreditcardId (PK)
- SensitiveId (PK, created by DME)

Now when we look into the data of the Table 1-3, 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. 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

Table 2: Metadata information after fragmentation

Table	Metadata
Customer	Normal
Membership	Sensitive
DME_creditcard	SensitiveDME
Customer_creditcard	Sensitive
DME_creditcard_senstive	SensitiveDME
DME_creditcard_mapper	SensitiveDME

Table 3: Segregated schema

Normal	Originally Sensitive	Sensitive DME
Customer	Membership	DME_creditcard
	Customer_creditcard	DME_creditcard_senstive
		DME_creditcard_mapper

The DME then moves the normal data to one database and originally ensitive 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))
  - Customer name
  - Customer address
  - Customer phone
  - Customer DOB

Bank DB-DME:
- Membership table containing
  - CustomerId (primary and Foreign Key(FK))
  - Password
  - Password question
  - Password answer
- Customer_creditcard table containing
  - CustomerId (primary key)
  - CardId (primary key)
- DME_creditcard_mapper table containing
  - Creditcard Id (PK)
  - SensitiveId (PK created by DME)
- DME_mapper table containing
  - Original table name (Combined PK)
  - New table name (Combined PK)

Server 2:
- DME_creditcard table
  - SensitiveId (PK created by DME)
  - Credit card no
  - Card expiry date

Server 3:
- DME_creditcard_senstive table (created by DME)
  - SensitiveId (PK FK created by DME)
  - CVV No.

The DME_mapper table is shown in Table 4. Now each database contains data which does not have

Table 4: DME mapper table

Original table name	New table name
Creditcard	DME_creditcard
Creditcard	DME_creditcard_senstive
Creditcard	DME_creditcard_mapper

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). Sensitiity = 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 x No. of critical tables

C2 = Cost of fragmentation of one sensitive table x No. of sensitive tables (this cost incurs only when required security is high)

C3 = Cost of creating a mapper table x (No. of critcal tables + No. of sensitive tables)

C4 = Cost of regeneration of fragmented tables x ([No. of critical table x No. of DME tables created newly] + [No. of sensitive table x 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:

$$C6 <= C1 + C2 + C3 + C4$$

C7 = Cost of encrypting/decrypting senstive 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 pocessing power the cost incured 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.

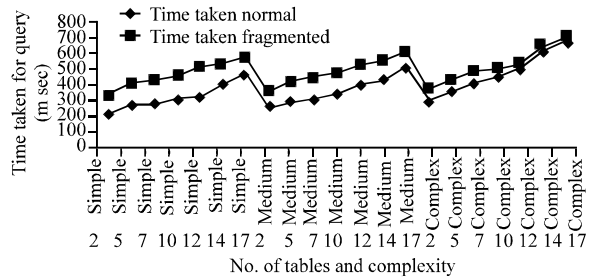

Fig. 2: Performance between normal and fragmented environment

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 research. 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. 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. Figure 2 shows the time taken for different type of queries in the normal and fragmented environment.

CONCLUSION

In this study, researchers 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 reserachers 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 refugee 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.

LIMITATIONS

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.

REFERENCES

BNA, 2009. Privacy and security law report, 8 PVLR 10, 03/09/2009. Copyright 2009 by The Bureau of National Affairs, Inc. (800-372-1033).

Fabre, J.C. and T. Perennou, 1995. Fragmentation of confidential bjects for data processing security in distributed systems. Proceedings of the 5th IEEE Workshop on Future Trends in Distributed Computing Systems, Auguast 28-30, 1995, Sheju Island, Korea.

Gibbs, M.R., G. Shanks and R. Lederman, 2005. Data quality, Database fragmentation and information privacy. Surveill. Soci., 3: 45-58.

Hose, K. and R. Schenkel, 2010. Distributed database systems-fragmentation and allocation. Cluster of Excellence MMCI. http://www.mpi-inf.mpg.de/departments/d5/teaching/ws10_11/dds/slides/DDS-2-print.pdf.

Sleit, A., W. Al-Mobaideen, S. Al-Areqi and A. Yahya, 2007. A dynamic object fragmentation and replication algorithm in distributed database systems. Am. J. Applied Sci., 4: 613-618.

Subashini, S. and V. Kavitha, 2011. A survey on security issues in service delivery models of cloud computing. J. Network Comput. Applic., 34: 1-11.

Wang, C., Q. Wang and K. Ren, 2009. Ensuring data storage security in cloud computing. Cryptology ePrint Archive, Report. http://eprint.iacr.org/.