A Data Partition Based Model to Enforce Security in Cloud Database

Osama M Ben Omran, Brajendra Panda
Department of Computer Science University of Arkansas Fayetteville, AR, USA

Abstract

Cloud computing has brought many advantages to companies and computer users. It allows different service providers to distribute many applications as services in an economical way. Therefore, many users and companies have begun using cloud computing. However, they are concerned about their data when they store it on a third party, the cloud. Fears of leakage of sensitive data or loss of privacy make the adoption of cloud services less attractive for organizations. In this paper an algorithm is presented to protect a table in a database from any leakage. We have developed a model with a view to offer secure data management capability in cloud databases. The model distributes and scatters the data over the cloud or data center in order to protect it from any leakage. Two models have been designed to divide a table by attributes relationship and by depending on non-sensitive and sensitive data. Also, this paper explains the idea of sending the entire domain of the sensitive table into the cloud. In addition, to increase security, the algorithm is designed to store each sensitive data in a different table with a different code and store this code at the client site. Furthermore, a new technique has been designed to collect the data from the cloud by using the bipartite matching algorithm to minimize load costs.

1. Introduction

With the potential to significantly decrease costs through optimization and increased operating and economic efficiencies, cloud computing is a great invention [2]. In addition, cloud computing could significantly improve its cooperation, agility, and scale, therefore enabling a truly global computing model over the Internet infrastructure [2]. Furthermore, the cloud computing with even higher performance has benefits in offering more scalable, fault-tolerant services [1]. Because of its high scalability, cloud computing offers unlimited computing resources on demand. This advance eliminates the need for the cloud service providers to plan far ahead on hardware provisioning [1]. Cloud computing has generated significant interest in industry, but it's still an evolving paradigm. Cloud computing attempts to combine computing technologies and the economic service model with the evolutionary development of several existing approaches, containing applications and spread services as well as information infrastructures consisting of groups of computers, networks, and storage resources [2]. Nevertheless, this potentially revolutionizing computing paradigm could become a huge failure without appropriate security and privacy solutions designed for the cloud [2]. There exist threats of unauthorized uses of the data by service providers and of theft of data from storage devices in the cloud. Furthermore, security is one of the most important concerns when moving to the cloud. Earning users' trust in the cloud providers occurs by providing the security of data in the cloud [3]. Organizations and individuals fear of leakage of their data especially the sensitive data when they put their data into the cloud. Because they typically result in data being present in unencrypted form on a machine owned and operated by a diverse organization rather than the data owner, current cloud services pose an inherent challenge to data privacy. There are threats of unauthorized users of the data by service providers and of theft of data from data servers in the cloud. Fears of sensitive data leakage or loss of privacy are a significant barrier to the adoption of cloud services. For example in 2007, criminals targeted the prominent cloud service provider Salesforce.com, and by a phishing attack succeeded in stealing customer emails and addresses. Furthermore, because of laws assigning geographical and other restrictions on the processing of personal and sensitive information by third parties, the use of cloud services as they are currently designed is constricted [4]. Even though organizations and individuals will save their money when they move into the cloud, they want to save more. They try to reduce the total amount of the expenses when they transfer the data from or to the cloud or data center. The total expense of the transfer depends on the workflow execution time, the total amount of data transmitted from the consumer to the storage resource, the total amount of data transferred from the storage resource to the consumer, and the storage used at the resource in terms of GB-hours [5].

The three key cloud delivery models are software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS). In SaaS, the cloud providers enable and provide application software as on-demand services. Therefore, we can use the provider's applications running on a cloud infrastructure and it is accessible from various client devices. PaaS enables programming environments to access and develop additional application building blocks. Such programming environments have an observable impact on the application architecture, such as constraints on which services the application can request from an OS [2]. In addition, consumer-created applications are installed on the cloud infrastructure using programming languages and tools supported by the provider.

Finally, In IaaS, the cloud provider contributes a set of virtualized infrastructural components such as virtual machines and storage on which customers can build and run applications. The application will eventually reside on the VM and the virtual operating system [2].

This paper provides a method to prevent information leakage. The model distributes and scatters the data over the cloud or data center in order to protect it from any leakage. It scatters the data depending on the relationship between the attributes and depending on non-sensitive and sensitive data. Also, it explains the idea of sending the entire domain of the sensitive table into the cloud. In addition, to increase security, the algorithm is designed to store each sensitive data in a different table with a different code and store this code on the client. Furthermore, a new technique has been designed to collect the data from the cloud by using the bipartite matching with Hungarian algorithm to minimize load costs. By applying this process, cloud service providers will reassure their customers and provide a high degree of transparency in their operations and privacy assurance. The organization of this paper is summarized as follows. First, it defines some related work. Second, it gives in details how the algorithm works and secures the data. Also, some examples have been provided to clarify the method.

2. Related Work

Cloud computing is a promising technology that presents an on-demand and large-scale computing infrastructure. Cloud computing refers to essential infrastructure for an up-and-coming model of service provision that has the advantage of reducing cost by sharing computing and storage resources. These new features have a direct impact on information technology budgeting but also affect traditional security, trust and privacy mechanisms [7]. It provides processing, storage, networks, and other fundamental computing resources, so the consumer is able to deploy and run arbitrary software, which can include operating systems and applications [6]. The secret data of individual users and companies is stored and managed by the service providers on the cloud which offers services on the other side of the Internet in terms of its users, and consequently results in privacy concerns [1]. This situation has existed for a long time in the computing literature, and several laws have been passed to protect users' individual privacy as well as business secrets. Nevertheless, because of a new relationship between users and providers, these laws have become out of date and inappropriate to the new scenarios [1]. Some research has been conducted in cloud database security to protect sensitive or non-sensitive information on the cloud. In papers [6] and [2] some issues have been discussed in cloud computing environments. A few papers have been published in cloud security database. As an example, in [3], researchers have described insider threat in cloud relational database systems. The paper explains how to develop and make knowledgebase in a cloud relational database system to monitor user activities and mitigate insider threats. Authors in paper [8] describe a model based on a client-based privacy manager in order to decrease users' fears of data leakage and loss of privacy. It uses the idea of employing obfuscation and de-obfuscation of data to reduce the amount of sensitive information held on the cloud. Authors in paper [9] explains how to prevent modification attacks on sensitive data items. They describe conditions to show how insiders can update data items maliciously in a relational database. The authors have offered two different methods that can be used to avoid modification attacks. Hiding dependencies among data items and denying write access to some data items are the two methods to prevent the attacks. Papers like [10] and [11] propose a probabilistic graphical model that can automatically infer true records and source quality in cloud data without any supervision. They leverage a generative process of two types of errors (false positive and false negative) by modeling two different aspects of source quality. Authors in paper [12] show how to protect the data files of a data owner in the cloud infrastructure by a set of security protocols, which are only accessible by a valid user. To protect the outsourced information, the paper explains how to combine access control and cryptography. It uses public key encryption for an access mechanism. Also, to prevent the trouble of key distribution and management, it proposes a modified Diffie-Hellman key interchange protocol between cloud service providers and the user for secretly sharing a symmetric key for secure data access. An additional paper [13] in cloud computing environments shows how to protect and ensure data confidentiality and fine-grained access control. The paper guaranteed data confidentiality by dividing and storing a data file into header and body. Also, it explains how a data owner can selectively decrypt the whole or part of the data using Type-based Proxy re-encryption. Authors in paper [14] use the vertical partitioning to divide the attributes of a relation or a record. It explains a new vertical partitioning algorithm using a graphical technique. The algorithm starts from the attribute affinity matrix that attributes are used together by transactions. The affinity matrix transforms into a complete graph and forms a linearly connected spanning tree. The algorithm generates all meaningful fragments in single repetition by considering a cycle as a fragment. In [15] the authors discuss new privacy and security concerns when users and companies give their data to external servers that then become responsible for their storage, management, and distribution. In addition to discussing these problems and concerns, the paper illustrated some developing directions introducing novel data protection approaches in outsourcing scenarios by data fragmentation and encryption. In [16], the paper introduces a cloud database module that provides database as a service. This paper developed the notion of cloud privacy and showed how using different levels of encryption layered as an onion can allow SQL queries to be processed over encrypted information.

3. Securing the Data Model

In this paper, the idea of dividing the table in a database into many tables has been used. The paper explains some new techniques to divide, distribute, and collect the data from the cloud. In general, the model is designed to work on two sites. First, at the client site, the algorithm and some secret data have been stored. Also, the algorithm has secret calculations and secret procedures to regenerate the user request from the cloud. Moreover, it has information about all tables in the database and their locations on the cloud. Second, at the provider site on the cloud, all tables in the database are stored on the cloud to make them available to various services across the Internet.

3.1. Dividing the Table depending on attributes relationship

Determining the sensitive attributes is achieved by the administrators or data owner. Sensitive attributes are the data that the administrators or data owner needs to hide from a provider or an attacker to prevent a privacy breach. Also, the administrators indicate which sensitive attributes cannot be in one group when the table is divided. This means if the algorithm puts these sensitive attributes together in one group, there is a high risk of privacy breach. Therefore, sensitive attributes are divided into difference tables based on this risk. Fig. 1 shows an example Sensitive Attribute Group. The two columns mean we cannot put sensitive attribute1 with the corresponding sensitive attribute2 when we divide the main table.

In this model, the study starts by mining the log file of the database system which we want to put on the cloud, and by this studying we get a statistical attributes matrix for all tables and all attributes. Fig. 2 shows the Statistical Attributes Matrix for all attributes in the table. This matrix describes the relationship between all attributes in the tables which we want to put on the cloud. It explains how many times or transactions the two attributes come together. If there is a number in the matrix between any two attributes, it means how many transactions accessed those attributes together. If the two attributes are sensitive and putting them together will leak sensitive data or cause a privacy breach, we have to separate them and put them in a different set or table on a different cloud or data center. If we put any two attributes together becoming a high risk of privacy breach, we do not need to count how many transactions accessed those attributes together. Depending on the number of transactions between two attributes and the sensitive attribute group, we build the statistical attributes matrix. By studying the number of queries between any two attributes in any table, we can apply the greedy algorithm which always makes the choice that looks best at the moment [17]. It will lead to a globally optimal solution. Optimal solution means that we can divide the table or the set into many tables or sets, so all attributes which have a maximum relation come together.

sensitive attribute1	sensitive attribute2
ID	EXPERIENCE
ID	SALARY
ID	NET_SALARY
NAME	EXPERIENCE
NAME	SALARY
NAME	NET_SALARY
SALARY	NET_SALARY

Figure 1. Sensitive Attribute Group

Attribute	ID (1)	NAME (2)	RANK (3)	NO_OF_DEPE (4)	EXPERIENCE(5)	SALARY (6)	NET_SALARY (7)
ID (1)		20	30	30	-	-	-
NAME (2)	20		30	30	-	-	-
RANK (3)	30	30		30	40	40	50
NO_OF_DEPE(4)	30	30	30		30	100	20
EXPERIENCE (5)	-	-	40	30		50	60
SALARY(6)	-	-	40	100	50		-
NET_SALARY (7)	-	-	50	20	60	-	

Figure 2. Statistical Attributes Matrix

Fig. 2 explains the number of transactions accessed those attributes together. If there are no numbers and only hyphen (-), this means the two attributes cannot be kept together for the security reason and must be separated depending on the sensitive attribute group table. Fig. 1 shows an example of the sensitive attribute group table, and it shows which attributes cannot set together. By looking at this matrix, we can know attributes {1, 2} cannot be stored with attributes {5, 6, 7} and attribute {6} cannot be stored with attribute {7}. Consequently, this table can be divided into three sets or three tables, and the possibilities are ({1, 2} and {5, 6} and {7}) or ({1, 2} and {6} and {5, 7}). In addition, we want to add the other attributes {3, 4} to one of the possibilities depending on the optimal solution. Therefore, we have to add the attributes {3, 4} to one of these five possibilities. In the next step, we apply the greedy algorithm to get the sets or sub-tables. We begin by get the result of all the possibility sets with their frequency or transactions number. Fig. 3 shows the result of Maximum Possibility Sets.

Operation no.	Possibility set	Attribute add	Frequency no.	Maximum	Max_Operation_no.
1	{1,2}	{3}	80	80	1
2	{1,2}	{4}	80	80	2
3	{6}	{3}	40	80	1
4	{6}	{4}	100	100	4
5	{6}	{5}	50	50	5
6	{7}	{3}	50	80	1
7	{7}	{4}	20	100	4
8	{7}	{5}	60	60	8
9	{5,7}	{3}	150	150	9

10	{5,7}	{4}	110	110	10
11	{5,6}	{3}	130	150	9
12	{5,6}	{4}	180	180	12

Figure 3. Result of Maximum Possibility Sets

To understand how to compute the Maximum Possibility Sets, let us consider operation number 1 in the Fig. 3. The possibility set is 1 and 2, and we want to add attribute 3 to this set.

- To calculate frequency no. we first check the number of frequency between 1 and 2 together in Fig.1 the Statistical Attributes Matrix and that equal 20 transactions. Also, we can check 3 with 1 is equal 30 transactions and 3 with 2 is 30 transactions, so the total is 20+30+30=80.
- To calculate the maximum value, we can get it by define which maximum frequency no. related to current attribute add is still the maximum value among all previous operation.
- To calculate max_operation_no value, we compute it by define which operation no. get the maximum frequency no. related to current attribute add among all previous operation.

After we apply all the possibilities sets and get the result of the maximum possibility sets, we apply the next algorithms on Fig. 3 to get the sets which have the maximum frequency together.

```
Input: arrays of the Result of Maximum Possibility Sets.
Output: the resulting tables Set.
1.  Initialize Set={}
2.  counter=max(operation no.)
3.  While counter>0
        o  If operation no.[counter]==
           Max_Operation_no[counter] and Attribute
           add[counter] ∉ Set
               ▪  Set= Set U (Possibility set[counter] U
                  Attribute add[counter])
        o  Counter--
4.  End while
5.  Return Set
```

After we apply the greedy algorithm from the previous steps, the results of the dividing table are ({1,2,3}, {4,5,6}, {7}) because they have the maximum frequency together. After we get the sets or the resulting tables, we can link them by encryption keys or develop an equation to link the resulting tables. The algorithm has been designed to encrypt and decrypt only the key of the table because of the complexity and cost to encrypt and decrypt the entire database.

3.2. Data Partition Depending on Sensitive Data

In this model an algorithm to protect sensitive data has been designed. The algorithm fragments and scatters sensitive data over the cloud in order to protect the data from unauthorized users. The technique will prevent the provider or attacker from discovering or reading private information from the cloud. It prevents the providers or attacker from linking the data to each other and from trying to get useful and real information, and it will be designed to work on client site and cloud site. The algorithm is used to divide a cloud table into a small group of tables. It stores each group in same location or in different locations on the cloud. The algorithm will divide the data into two groups: non-sensitive attributes and sensitive attributes. It will use conflict table to explain which sensitive attributes cannot become in one group when the cloud table is divided. Therefore, sensitive attributes are divided into other sensitive attribute groups depending on the conflict table. Finally, all non-conflict sensitive attributes are stored with one of conflict sensitive attribute, which has maximum query with it as defined by either mining log file or using the Apriori Algorithm. In addition, all resulting tables at the end are linked to each other by keys. We can generate unique random numbers and send them with the resulting tables into the cloud, and these numbers are used to link the data in the resulting tables. These numbers must keep and secure at the client site, and each related numbers are stored together to be known. We can add a coordinator in the middle between the client site and the cloud to store these numbers and the information of the dividing tables if we want to reduce the workload of processing on the client site. Also, we can use this coordinator to do all of the workload of processing the final query. Fig. 4 shows the idea or resulting of non-sensitive and sensitive attributes.

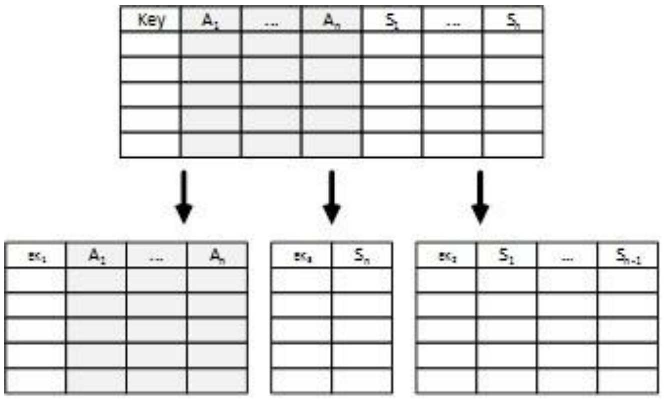


Figure 4 the Result of the Dividing table.

The model is designed to store all the resulting tables in one data center on the cloud and this will improve the performance and still secure. On the other hand, we can use a technique to distribute and scatter the resulting tables over many data centers to improve the security. Therefore, the algorithm will store all resulting tables on different data centers. The system will have a number of data centers on the cloud, which are used to store the resulting tables on them. The system may have many data centers in different locations depending on the cloud. Table 1 shows the location of the data centers. Based on the structure of the original table, the algorithm will store each resulting table depending on the rank of the first sensitive attribute. The rank is the number of the field in the main table, and it will use the following equation.

$$D = (R) \bmod (M) \qquad (1)$$

Where D is the data center number on the cloud, R is the rank of the attribute or number of the attribute in the structure of the record in the table, and M is total number of the data center on the cloud. If the key consists of more than one attribute, the algorithm will assign a unique number to each record as a key and store that in a new attribute. The information of the data centers is stored on the client site and must be secure. The location may be in one provider or different providers. This information is organized as shown in table 2 which called a metadata table. The information of the metadata table is stored on the client site and must be secure. After the design stage finishes and the final tables are on the cloud, the system is now ready to run the query for the end user. Depending on the query and the dividing of the attributes on the cloud, the algorithm processes and modifies the query on the client site. After that, it sends the modified query to each server or cloud, and the provider on the cloud processes the query and returns the result to the client. The client assembles all the queries on the client site and produces the final result. As we explained before we can use a coordinator to do all processing for the client and just give the right result to the client.

Table 1. The location of the data centers.

Data center	Location
0	…
…	…
m-1	…

Table 2. The structure of the metadata table.

Original table name	Dividing Table	Attribute Name	Location of the Data Center	Rank of the attribute
…	…	…	…	…

1. *An Example Scenario to Divide the Data Depending on Sensitive Data*

   We introduce a simple example to show how the algorithm will divide a given table into many different tables and after that the algorithm stores the resulting tables on the cloud.

A. At Design stage

For example, the structure of the table, the sensitive table, and conflict table are given as explained:

The table= Emp_salary={ID, Name, Rank, # of Dependent,Base_salary, Experience, Tax, Salary, Net_Salary}.

Sensitive Table	Rank
Experience	6
Salary	8
Net_Salare	9

Conflict Table	
Attribute 1	Attribute 2
Salary	Net_Salary

Secondly, the algorithm separates the attributes into non-sensitive table and sensitive table, so we get:

T1 = {ID, Name, Rank, # of Dependent, Base_salary, Tax}
T2 = {Experience, Salary, Net_Salary}

Thirdly, depending on conflict table, the algorithm divides T2 and puts each conflict attribute in different tables. So the result will be:

T2 = {Salary}, T3 = {Net_Salary}

Next, the algorithm will search and mine the log file or use Apriori Algorithm for the attributes, which are sensitive and are not found in conflict table. The algorithm will put these attributes with the most frequent of these attributes that come in query with the sensitive attributes which is found in conflict table. For example, if the algorithm finds the most frequent of the Experience attribute comes with the Salary attribute in the queries, the algorithm will place the Experience attribute with the Salary attribute. So the result will be:

T2 = {Salary, Experience}, T3 = {Net_Salary}

In the next step, the algorithm will generate random numbers for each sensitive table and send them with the resulting tables. These numbers are used to link the data in the resulting tables. After these calculations, the final result will be:

T1= {Key or random number, Name, Rank, # of Dependent, Base_salary, Tax}
T2= {Random number 2, Salary, Experience}
T3= {Random number 3, Net_Salary}

After that the algorithm will get the location of the Data Center, which is used to store the tables, by using the following equation:

Data center number= (Attribute rank) mod (m)
L1 = 1 mod m, L2 = 8 mod m, L3= 9 mod m

The information about the metadata table is as shown in table 3.

Table 3. The location of the data centers.

Original table name	Dividing Table	Attribute Name	Location of the Data Center	Rank
Emp_salary	T1	ID,Name, Rank,#of Dependent, Base_salary, Tax	L1	1
Emp_salary	T2	Salary, Experience	L2	8
Emp_salary	T3	Net_Salary	L3	9

B. At Process and Management stage

After the tables are stored on the cloud, the tables are now ready to be accessed by users. For example, assume that a user requests the following query:

Select ID, Name, Salary, Net_Salary from Emp_salary Where ID=”1112”

Then, the query will be modified by the query processor depending on the dividing tables, and it will generate the following SQL statements and send them to the cloud:

Select ID, Name from t1 Where ID=”1112”
Or we can use random number:
Select ID, Name from t1 Where ID=”11445560”
Select Salary from t2 Where Random number 2 =”77808091”
Select Net_Salary from t3 Where Random number 3 =”11135577”

Where ” 11445560”, ” 77808091”, ”11135577” are random numbers which have been stored on the cloud with the tables.

2. *Observation*

There are some important observations on the algorithm as the following explains.

Observation 1. To prevent privacy breach when the algorithm requests data from more than one table related to each other, we must do one of the next two points: first, the algorithm must return one record from one table and many records from the other tables. The number of return records is defined by the threshold from the administrators to prevent a privacy breach. In this case, we must have different sensitive values equal to the threshold value, so the probability to get the sensitive values is:

$$\text{Prob (get sensitive value)} = 1/\text{Threshold} \qquad (2)$$

Second point, to get high security, the algorithm must return many records from all tables depending on the number of the threshold. In this case, we must have different sensitive values equal to the threshold value and different non-sensitive values (m) equal to the threshold value, so the probability to get the sensitive values is:

$$\text{Prob (get sensitive value)} = 1 / (\text{Threshold})^m \qquad (3)$$

Observation 2. If a provider or an attacker only gets access to non-sensitive attribute D1, where $D1 \in R$ and R is a record in a table in a database DB, then the provider or the attacker cannot get access to secret information, and we still have confidentiality. Also, we do not have any privacy breach. For example, if the provider gets access to only table T1 which is non-sensitive attributes,
T1= {ID, Name, Rank, # of Dependent, Base_salary, Tax}, the provider cannot link and get any sensitive information because the table does not have any sensitive information.

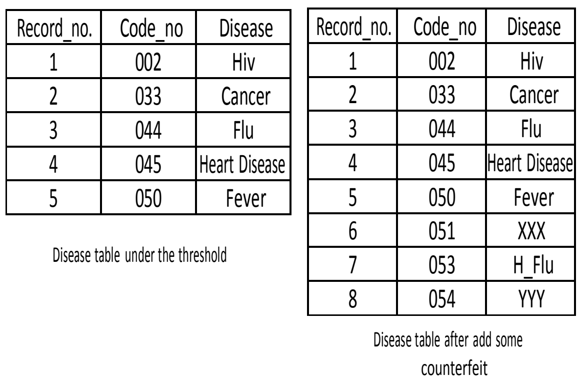
Observation 3. If a provider or an attacker only gets access to sensitive attribute D2, where $D2 \in R$ and R is a record in a table in a database DB, then the provider or the attacker cannot get access to secret information, and we still have confidentiality. Also, we do not have a privacy breach. For example, if the provider gets access to only table T2 which is sensitive attributes,
T2= {Random number, Salary, Experience}, the provider cannot link this sensitive information to anyone because the table T2 does not have any information about any person. Also, if the provider gets the two attributes, he cannot use them together to link to anyone.

Observation 4. If a provider or an attacker gets access to non-sensitive attribute D1 and sensitive attribute D2, where D1,D2 are parts of two records in different tables, then the provider or the attacker can get access to secret information, and we do not have confidentiality. Also, we will have a privacy breach. For example, if the provider gets access only to table T1 which is non-sensitive attributes and to table T2 which is sensitive attributes,
T1= {ID, Name, Rank, # of Dependent, Base_salary, Tax},
T2= {ID, Salary, Experience}, the provider can link the sensitive information because both tables will have the same ID for a particular person.

Observation 5: If a provider or an attacker gets access to sensitive attribute D1 and D2, where $D1, D2 \in R$ and R is a record in a table, and D1 and D2 together more than the probability of get sensitive value, then the provider or the attacker can get access to secret information, and we do not have confidentiality. Also, we will have a privacy breach. For example, if the provider gets access only to table T4 which is sensitive attributes,
T4= {ID, Age, Address, Nationality, Disease}, the provider or an attacker can use these attributes to link the disease to a person. For example, if one person has the same age, same address, and same nationality, the provider can link this disease to this person.

C. *Add All Possibilities of the Sensitive Data*

In this stage, after we divide the table and get the sets of attributes, we send all the sets or tables to the cloud. The tables of the sensitive data are sent with all possibilities of the sensitive data which means the entire domain. The number of the possibilities of the sensitive data must be above the threshold. If the number of the possibilities of the sensitive data is below the threshold, we have to add some counterfeit (dummy) records into the data set to protect the data from any leakage. For example, consider the diseases table which is a sensitive table shown in Fig. 5, and the table contains all possible domains of the diseases. If the number of diseases or number of records in the table is below the threshold, we have to add some different counterfeit records to the table until we satisfy the threshold. The reason for that is the algorithm will return number of records equal to the threshold from the sensitive table “Disease” to protect the data from any leakage when the user will have requested data from the “Disease” table. At the client, the algorithm will identify the right record and submit to the user. Fig. 5 shows an example of adding counterfeit records to the “Disease” table where the threshold is eight.



Record_no.	Code_no	Disease
1	002	Hiv
2	033	Cancer
3	044	Flu
4	045	Heart Disease
5	050	Fever

Disease table under the threshold

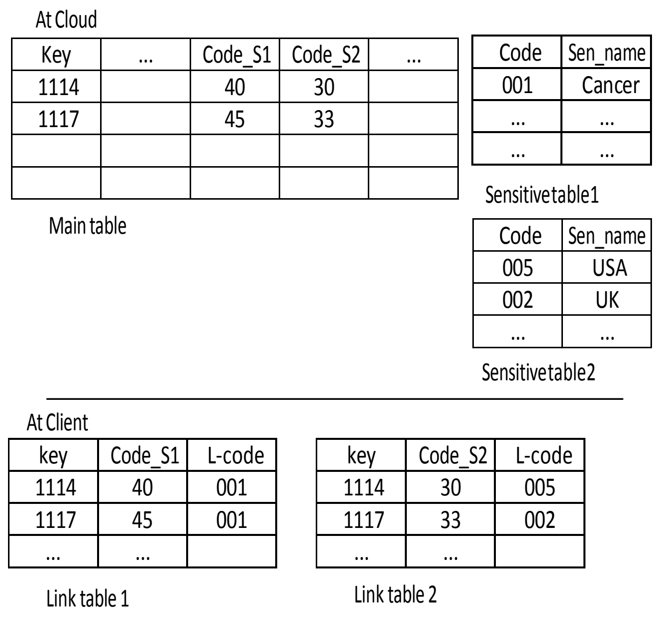
Record_no.	Code_no	Disease
1	002	Hiv
2	033	Cancer
3	044	Flu
4	045	Heart Disease
5	050	Fever
6	051	XXX
7	053	H_Flu
8	054	YYY

Disease table after add some counterfeit

Figure 5. Add Counterfeit records to table

D. Coding the Sensitive Table

After we divide the table into some partitioning tables vertically and add all the possibilities of the entire domain and add some fake records based on the threshold, the module is design to store them on the cloud. To increase the security, we can store the resulting tables on different data centers or different clouds. Therefore, the main data and the sensitive data are stored on the cloud.



At Cloud

Key	...	Code_S1	Code_S2	...
1114		40	30	
1117		45	33	

Main table

Code	Sen_name
001	Cancer
...	...
...	...

Sensitivetable1

Code	Sen_name
005	USA
002	UK
...	...

Sensitivetable2

At Client

key	Code_S1	L-code
1114	40	001
1117	45	001
...	...	

Link table 1

key	Code_S2	L-code
1114	30	005
1117	33	002
...	...	

Link table 2

Figure 6. Coding the Sensitive Tables

In the next step, we generate a code for each sensitive data, so each sensitive data will have a different code. This code is used to link the sensitive tables from the cloud to other tables called link tables on the client site. Fig.6 shows how to link these tables by the attributes called Code and L-code. For security reasons, the link tables are stored on the client site. In addition, the link tables are used to link the information between the main tables and the sensitive data on the cloud. To hide the original code of the sensitive data, we use another code like Code_S1 to link the main and link tables. The idea is to give different code for the same sensitive data, and we use this code to connect the link and main tables. Therefore, when the client enters new data into the main table, the algorithm generates a new code for the sensitive data and sends it to the cloud. Fig. 6 shows coding the Sensitive Tables.

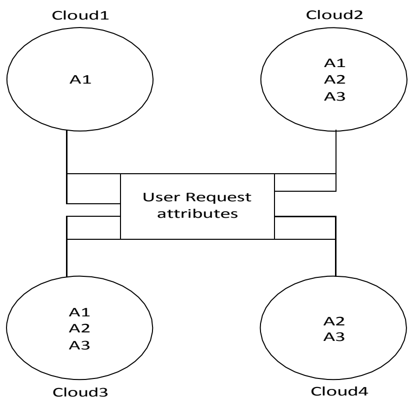


Figure 7. Distributing Data over the Clouds

E. Getting Data from the Cloud by Lowest Cost

After we divide the table into some tables and distributed the tables on different data centers in the cloud, a new technique has been designed to collect the data from the different data centers by using bipartite matching with Hungarian algorithm to minimize load costs. As explained previously, the total cost is dependent on the total amount of data transmitted from the consumer to the storage resource and from the storage resource to the consumer. Fig. 7 shows different tables with different attributes distributed over four different data centers on the cloud.

Attribute name	Cost	Cloud
A1	0.05	C1
A1	0.10	C2
A1	0.20	C3
A2	0.04	C2
A2	0.09	C3
A2	0.20	C4
A3	0.03	C2
A3	0.09	C4
A3	0.10	C3

Figure 8. Cost Table to load each attribute from different cloud

1. An Example Scenario to Get the Data from the Cloud at the Lowest Cost

We introduce a simple example showing the algorithm will work to get the data from the cloud at minimum cost. Let us say we have four clouds or data centers, and the data has been distributed as shown in the fig. 7. The user requests the data {A1, A2, A3}. They are sensitive, and the algorithm cannot get them from one data center for security reasons. The idea of the algorithm is designed to allocate a cost table on each client as shown in the fig. 8. Therefore, when the user requests a query from the cloud, the query will be modified by the query processor depending on the lowest cost after applying bipartite matching algorithm as shown in the fig. 9. Consequently, the algorithm will generate more than one query based on the partitioning of the data which the user wants. Fig. 9 shows how the algorithm applies the bipartite algorithm where A1, A2, and A3 are the data which was requested by the user, and C1, C2, C3, and C4 are the clouds or data centers which hold the data. Based on this example, the algorithm will request A3 from C2 where the cost is 0.03 units, A2 from C3 where the cost is 0.09 units, and A1 from C1 where the cost is 0.05 units. Therefore, the total is 0.17 units.

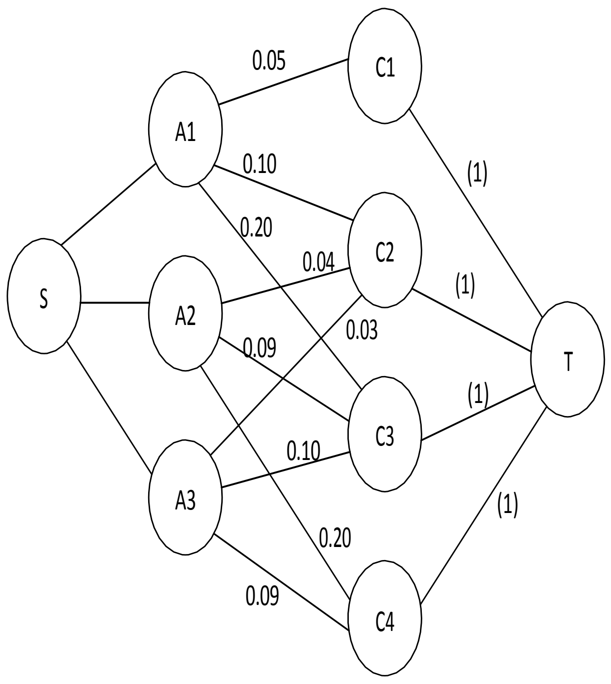


Figure 9. Graph for Bipartite Matching Algorithm

4. Conclusion

Numerous companies and computer users need to protect their data when they move and manage data on the cloud. This paper has discussed how to protect information on the cloud. It has explained how to protect a table in a database by dividing it into many tables and storing them on the cloud to protect them from any leakage. An algorithm has been provided to do this work. We have developed a model with a view to offer secure data management capability in cloud databases. It scatters the data depending on the relationship between the attributes and depending on sensitive data. It applies greedy algorithm to get the best division for the table. Also, another model to distribute the data depending on non-sensitive and sensitive data has been explained. This paper has also explained the idea behind sending the entire domain of the sensitive table into the cloud, and storing each sensitive data on a different table with a different code, which is stored on the client site. In addition, to minimize the load cost, a new technique has been designed to collect the data from the cloud by using the bipartite matching algorithm. Finally, some example scenarios have been explained to show how the algorithm works.

5. References

[1] Minqi Zhou, Rong Zhang, Wei Xei, Weining Qian, and Aoying Zhou, "Security and Privacy in Cloud Computing: A Survey," Sixth International Conference on Semantics, Knowledge and Grids, pp. 105- 112, November 2010.

[2] Hassan Takabi, James B.D. Joshi, and Gail-Joon Ahn, "Security and privacy challenges in cloud computing environments," IEEE Computer and Reliability Societies, pp. 24-31, November/December 2010.

[3] Qussai Yaseen and Brajendra Panda, "Tackling Insider Threat in Cloud Relational Databases," IEEE/ACM Fifth International Conference on Utility and Cloud Computing, pp. 215-218, 2012.

[4] Siani Pearson and Azzedine Benameur, "Privacy, security and trust issues arising from cloud computing," 2nd IEEE International Conference on Cloud Computing Technology and Science, pp. 693-702, 2010.

[5] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman, and John Good, "The Cost of Doing Science on the Cloud: The Montage Example," Proceedings of the 2008 ACM/IEEE conference on Supercomputing, p. p. 50, 2008.

[6] Hyun-Suk Yu, Yvette E. Gelogo, and Kyung Jung Kim, "Securing Data Storage in Cloud Computing," Journal of Security Engineering, pp. 251-259, June 2012.

[7] Siani Pearson and George Yee, Privacy and Security for Cloud Computing. London: Springer, 2013.

[8] Mowbray Miranda and Siani Pearson, "A client-based privacy manager for cloud computing.," Proceedings of the fourth international ICST conference on COMmunication system softWAre and middlewaRE.ACM, pp. 1-8, 2009.

[9] Qussai Yaseen and Brajendra Panda, "Malicious Modification Attacks by Insiders in Relational Databases: Prediction and Prevention," IEEE Second International Conference on Privacy, Security, Risk and Trust, pp. 849-856, August 2010.

[10] Sean Thorpe, "A theoretical model for compiling truthful forensic evidence from the hypervisor log cloud database environment," 2013.

[11] Bo Zhao, Benjamin I. P. Rubinstein, Jim gemmell, and Jiaw Han, "A Bayesian approach to discovering truth from conflicting sources for data integration," pp. 550-561, August 2012.

[12] Sunil Sanka, Chittaranjan Hota, and Muttukrishnan Rajarajan, "Secure Data Access in Cloud Computing," In Internet Multimedia Services Architecture and Application (IMSAA), 2010 IEEE 4th International Conference, pp. 1-6, 2010.

[13] Jeong-Min Do, You-Jin Song, and Namje Park, "Attribute based Proxy Re-Encryption for Data Confidentiality in Cloud Computing Environments," In Computers, Networks, Systems and Industrial Engineering (CNSI), 2011 First ACIS/JNU International Conference, pp. 248-251, 2011.

[14] Shamkant B. Navathe and Minyoung Ra, "Vertical Partitioning for Database Design: A Graphical Algorithm," ACM SIGMOD Record , pp. vol.18 no.2 440-450 , 1989.

[15] Pierangela Samarati and Sabrina De Capitani di Vimercati, "Data protection in outsourcing scenarios: Issues and directions," In Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, pp. 1-14, 2010.

[16] Carlo Curino et al., "Relational cloud: A database-as-a-service for the cloud," 5th Biennial Conference on Innovative Data Systems Research, CIDR 2011, pp. 235-240, January 9-12 2011.

[17] Thomas H Cormen, Charles E Leiserson, Ronald L. Rivest, and Clifford Stein, Introduction to Algorithms, 3rd ed. London, England: MIT Press, 2009.