Improving Database Security using DD-PLAC architecture

B.A.Kelkar [1] , Priyanka Soude[2] , Swati Shelake[3] , Shambala Mali[4] , Pallavi Patil[5] , Mrunali Sanger[6] .

[1,2,3,4,5,6]Computer Science and Engineering, Sanjay Ghodawat Group of Institutions,Atigre.(India)

***Abstract*-Cloud computing is a technology for sharing of resources on the internet. Confidentiality of data storage is very essential for assurance of data security. However, existing cloud computing data storage methods do not provide robust data privacy. Cloud data is often stored in plain text Very few companies have an absolute understanding of the sensitivity levels their data. The online applications are also vulnerable to attacks and can gain access to the sensitive data as the attacker can easily exploit software bugs. Today's the biggest challenge in cloud computing is security and privacy of the sensitive data. Several techniques are being developed for improving storage services, but the data confidentiality and privacy solutions for cloud database are still in research. This paper mainly focuses on the security of SQL database using DD-PLAC architecture which provides confidentiality of the data stored on cloud. Here we combine encryption of data with SQL operations to ensure data privacy and confidentiality.**

Keywords-DD-PLAC Proxy-less architectures, Symmetric Key Algorithm (SKA), access control, Security and Privacy for cloud database.

1. INTRODUCTION

Confidential and critical information is stored across the cloud mostly through third-party. The third- party acts as an intermediate agency. If third party identified the pattern of security mechanism which leads to insecurity of stored information. Ensuring the confidentiality of information requires the best data management choices like – only authorized parties can access the original plain text. In this paper, we propose a solution that removes the third-party involvement. The existing architecture for ensuring data privacy on cloud is Proxy Less architecture (PLAC), in which the proxy server between the client and database is removed. The PLAC architecture has two problems. First it is prone to single point failure (means that data will be lost if the system failure occurs due to some problems) and second is bottleneck (i.e. if too many requests are coming for same operation, the system may not be able to process most of the requests). Due to these drawbacks the performance of cloud is degraded.

To overcome these issues the DD-PLAC (Distributed Database-Proxy Less Architecture) is proposed. In this architecture, distributed cloud database is used where data is distributed across the cloud. The benefit of storing on distributed database is decrease in load on single database. The access permission is given to each authorized user according to their roles. The DD-PLAC architecture is implemented with the help of vertical fragmentation of data. In vertical fragmentation the table is fragmented according to the columns. Fragmentation overcomes the problem of bottleneck. In this architecture data is stored in encrypted format using the asymmetric key algorithm. RSA algorithm is used with help of Hash function. The encryption key is produced based on the clients information. This DD-PLAC architecture supports the concurrent execution of the operations on the encrypted data and does not introduce any intermediary proxy server between the client and cloud database. By eliminating the proxy servers, it achieves the scalability and reliability etc.

2. RELATED WORK

In literature, following types of architectures are defined to preserve the privacy:

1. Proxy-based architectures
2. Proxy-less architectures that store metadata in the clients.
3. Proxy-less architectures that store metadata in the cloud database.

1. Proxy-based architectures (PBA):-

The proxy-based architectures are prone to performance degradation because the proxy is a bottleneck and a single-point-of-failure that limits availability, scalability and elasticity of the cloud DBaaS (Database as a Service). Since the proxy must be trusted, it cannot be outsourced to the cloud and has to be deployed and maintained locally. Moreover, proxy-based architectures cannot scale trivially by increasing the number of proxies. Such a naive solution would imply the replication of metadata among all the proxies, but this would require synchronization algorithms and protocols to guarantee consistency among all the proxies.

2. Proxy-less architectures that store metadata in the clients (PLA):-

The Proxy-less architecture that store metadata at client side does not use an intermediate proxy. So the clients can connect directly to the cloud database, this architecture provides availability, scalability and elasticity. So, each client has its own encryption engine and manages a local copy of metadata. So, this solution can represent a sub-case of the proxy-based architecture, in which a different proxy is deployed within each client. A similar architecture for cloud accesses would suffer from the same consistency issues of proxy-based architectures.

3. Proxy-less architectures that store metadata in the cloud database (PLAC):-

The third architecture is proxy-less architectures that stores metadata in the cloud database. In this the metadata is stored to the cloud database, but the encryption engine is executed by each client. As metadata are not shared among all the clients there is no need of synchronization. Client machines execute a client software component that allows a user to connect and issue queries directly to the cloud DBaaS. This software component retrieves the necessary metadata from the untrusted database through SQL statements and makes them available to the encryption engine at the client. Multiple clients can access the untrusted cloud database independently, with high availability, scalability and elasticity. The drawback of this architecture is bottleneck and the single point of failure.

3. THE DESIGN OF DD-PLAC

A client organization having trusted Database Admin machine hosts the DD-PLAC client for the creation and management of the encrypted database. All database users can issue SQL operations directly to the cloud database even from geographically distributed locations by executing a DD-PLAC client on their machines. The entire set of data is stored in an encrypted form in the cloud database. Due to the SQL-aware encryption strategies, the cloud database engine can execute queries on encrypted data without accessing any decryption keys. Even metadata that are necessary to manage encryption strategies are considered critical information, hence DD-PLAC stores them encrypted in the cloud database: the Database Admin and the users can efficiently retrieve metadata

through standard SQL queries. It is assumed that Database admin is the only who has root credentials for the client application and no internal or external attackers able to access, or crack the credentials.

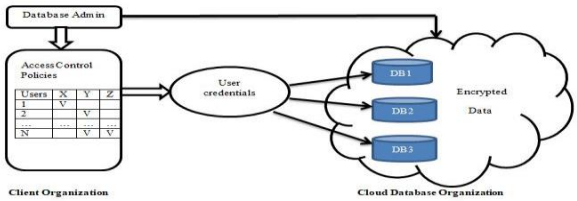

Figure 1: DD-PLAC Architecture

For implementation of PLAC architecture, for every record, different encryption key is used and the encryption key, and unique record id is stored again in encrypted format in the metadata. Using this metadata it is easier to find the encryption key of record being stored in database for the application. Multiple users operation do not affect to the structure of the encrypted database. This encryption algorithm ensures the data privacy and the distributed cloud database will ensures the continuous service from the cloud service provider as well as confirms the data is safe in hands of cloud database.

4. IMPLEMENTATION DETAILS

4.1 The implementation of DD-PLAC architecture we create one commercial application and cloud is used to deploy the application and this cloud provides the database storage as MySQL server.

The DD-PLAC architecture is divided into following four parts:

1. Distributed database creation
2. Implementation of algorithm for data encryption and decryption
3. Metadata generation
4. Developing Web-application

1. Distributed database creation:-
Distribute the database over the network by using vertical partitioning technique.

2. Implementation of algorithm for data encryption and decryption:-
For maintaining security in application, AES symmetric key algorithm is used and based on the information entered by the user, Encryption key is generated.

Encryption Algorithm

Input: Plaintext data P,SKA,512 bit hash function H

Output: Encrypted data C

Steps:

1. Read input from user as Plaintext Data P.
2. Generate random key R from the personal information of user.
   R= (Personal info)512
3. By applying hash function generate symmetric key.
   K=H(R)
4. Encryption of data by using the symmetric key K.
   C=SKA (P,K)
5. Store the encrypted data in table format.
6. Store the symmetric key K and random key R in metadata in encrypted format by using the encryption of metadata.

Decryption Algorithm

Input: Encrypted data C, SKA, 512 bit hash function H

Output: Plaintext data P

Steps:

1. Retrieve the symmetric key K from the metadata.
2. Decrypt the stored encrypted data by using the symmetric key K.
   P=SKA(C, K)
3. Get the original plaintext data P and do the operations on the data.

3. Metadata generation:-

Metadata stores the Unique Registration Id and Encryption Key in encrypted format.

4. Developing Web-application:-

The web application is developed using PHP, includes the following details of human being:

1. Personal information.

2. Banking details.

3. Policy details.

4.2 Generation of Secret Key

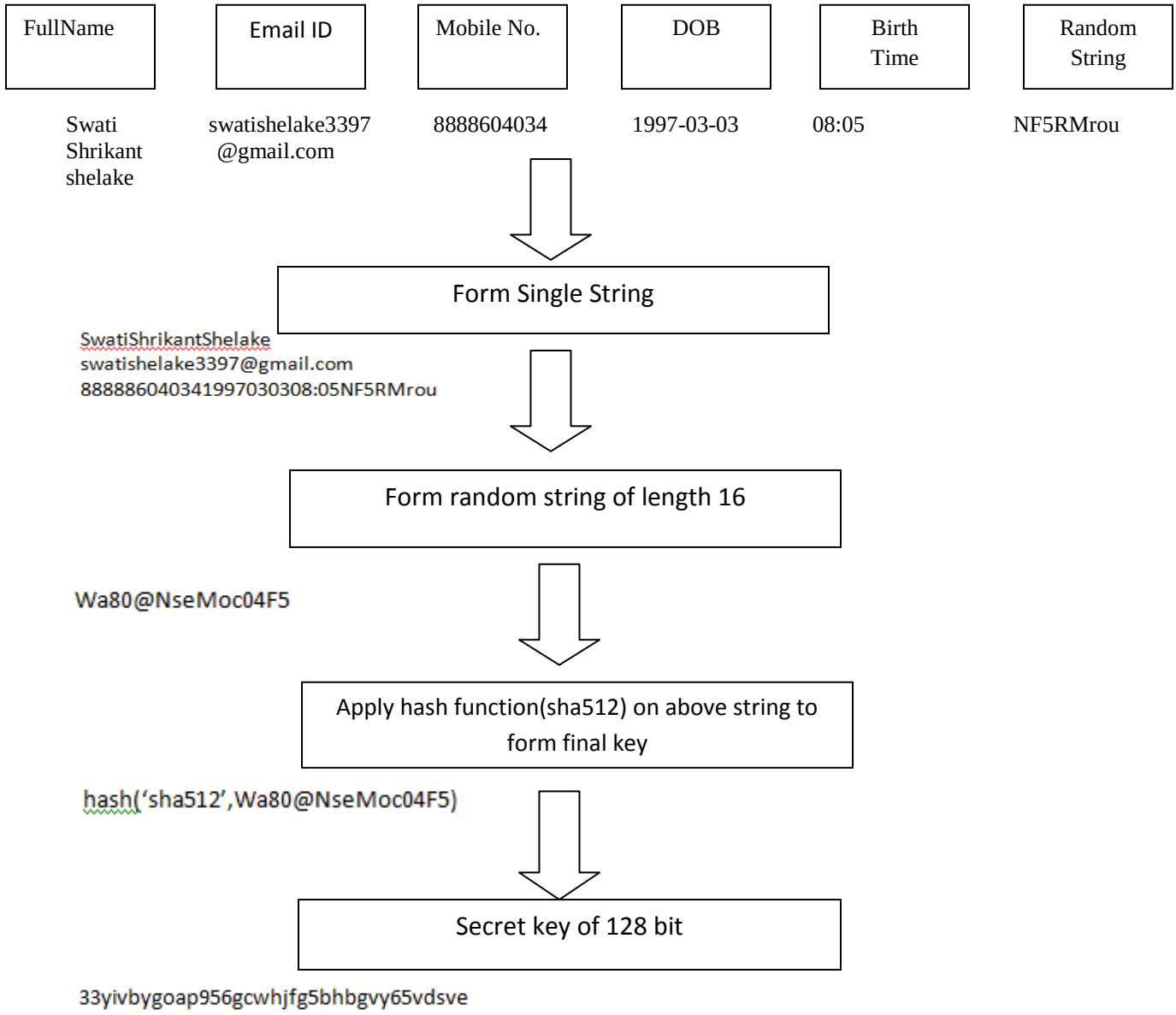

Figure 3: Secret Key Generation

4.3 Queries over encrypted data:-

We can execute the queries on distributed database same like as DBMS.DD-PLAC architecture supports to execute operations such as join, aggregation, projection and selection etc. on cipher text.

DD-PLAC architecture performs following procedure when perform queries on database:-

1. User can perform operation through the application on encrypted data.

2. This data is forwarded to DBMS server. This server interprets the requested data.

3. Finally DBMS server sends the resultant data to user.

5. RESULT ANALYSIS

5.1. Performance analysis:-

In a regular database if multiple users perform operations simultaneously on same database, performance degradation can be observed. To avoid this, we use distributed database. DD-PLAC architecture refers distributed database on cloud in which vertical fragmentation is done on tables to handle performance related problems. If multiple users perform an operation it does not affect system performance. Multiple replicas are stored on the cloud server, so there is no chance to loss of data. User can access the data anytime anywhere resulting into fast data access.

Following figure 5.1.1 shows the access time required to data using Distributed database:-

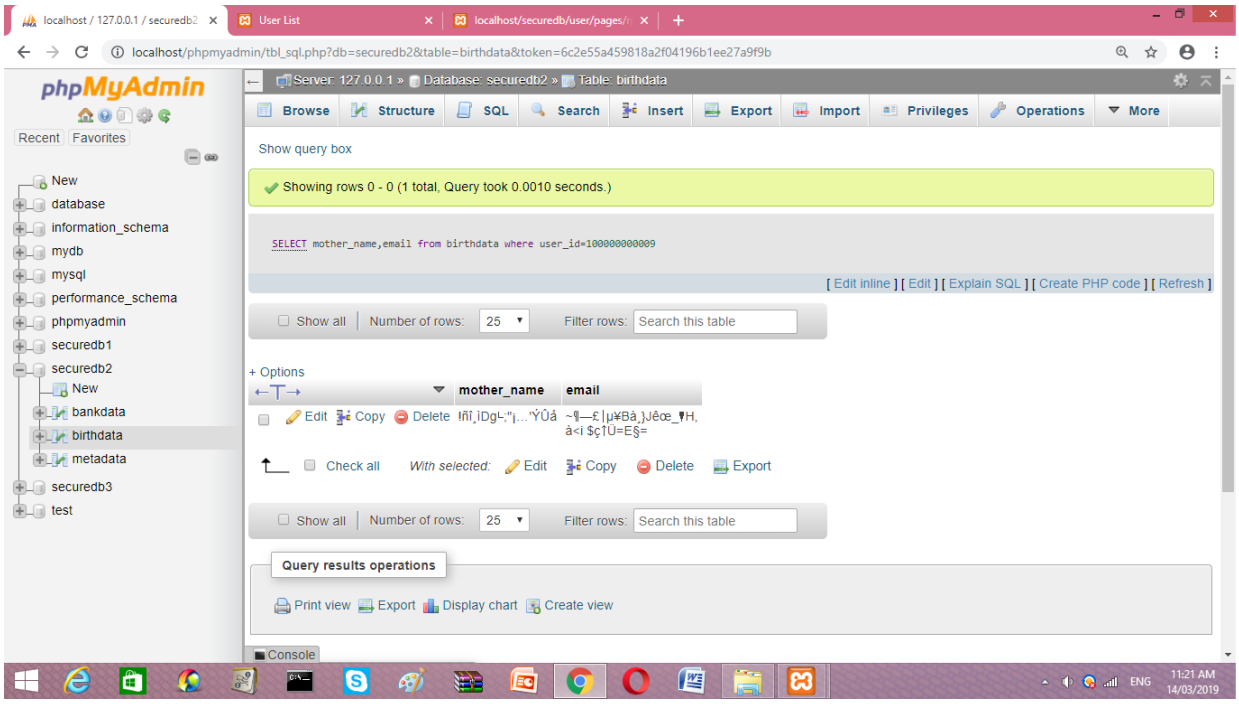

Figure 5.1.1 : access time required to data using Distributed database

5.2. Security analysis:-

Security of data is maintained using AES (Advanced Encryption Standard) algorithm. AES algorithm is combined with hash function to provide strong level security to data. AES algorithm performs encryption on given input and generates the cipher text. Hash function is used to generates the key which is formed by choosing random characters from given input. So the resultant secret key is very strong because of this level of encryption is also increased.

6. CONCLUSION

So the proposed architecture of distributed database refers proxy less architecture. In this architecture there is no any server in between client and server. The main advantages of this architecture are Data Security, Data availability, Data consistency. The DD-PLAC architecture provides a strong level of security and privacy. All the data which is stored on the cloud provider are encrypted through cryptographic algorithms which allow the execution of standard SQL queries on encrypted data. This architecture is also provides direct, independent and concurrent access to the cloud database. It does not rely on a trusted proxy that represents and also avoids the single point of failure and a system bottleneck, which in turn increases the availability and scalability of cloud database services.

7. REFERENCES

[1] Luca Ferretti, Michele Colajaani, and Micro Marchetti, " Distributed, Concurrent, and Independent Access to Distributed Database," 2015.

[2] Amjad Alsirhani, Peter Bodorik, Srinivas Sampalli, "Improving Database Security in Cloud Computing by Fragmentation of Data," 2017.

[3] Sajjan R.S., Vijay Ghorpade, Vishvajit Dalimbkar, "A Survey Paper on Data security in Cloud Computing," 2016.

[4] Parneet Kaur and Sachin Majithia, "Various Aspects for Data Migration in Cloud Computing and Related Reviews," 2014.

[5] L. M. Kaufman, "Data Security in the World of Cloud Computing," 2009.

[6] A. Hudic, S. Islam, P. Kieseberg, S. Rennert, and E. R. Weippl, "Data Confidentiality using Fragmentation in Cloud Computing," 2013.

[7] J. Daemen, "The design of Rijndael : AES - the Advanced Encryption Standard with 17 Tables," 2002.

[8] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R. Motawani, "Distributing Data for Secure Database Services," 2011.

[9] Danie J. Abadi, "Data Management in the Cloud:Limitations and Opportunities," 2009.

[10] "Addressing Data Security Challenges in the Cloud A Trend Micro White Paper," 2010.

[11] James Broberg, Rajkumar Buyya, Zahir Tari, "MetaCDN:Harnessing Storage Clouds or high performance content delivery," 2009.