International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 02, February 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

# A SURVEY ONPROVIDING DATA CONFIDENTIALITY AND SECURE ACCESS TO UNTRUSTED CLOUD DB

**Minaj Mulla[1], Nusrat Inamdar[2]**

*[1]Assistant Professor, [2]Student, Dept. of Computer Science Engineering,*
*S.I.E.T College of Engineering and Technology, Vijaypur, Karnataka, (India)*

**ABSTRACT**

*Placing the data in the cloud should come with the guarantee of security, confidentiality and availability of the data at all the time. several alternatives exist for the storage services, while data confidentiality solutions for the database as a service model are still have scope to work. In the paper, the architecture represents cloud database services with the data confidentiality and also possibility of executing the concurrent and independent operations on encrypted dataand .This is the solution which supports geographically distributed users to directly connect to cloud database and to execute independent operations and also provide a supervisor on N Number of clients. The proposed architecture has the further advantage of eliminating any intermediate server between the cloud client and the cloud provider.*

***Keywords*: Cloud, Security, confidentiality, Secure DBaaS, database**

## I. INTRODUCTION

In a cloud context, where critical information is placed in infrastructures of untrusted third parties, ensuring data confidentiality is of paramount importance. This requirement imposes clear data management choices: original plain data must be accessible only by trusted parties that do not include cloud providers, intermediaries, and Internet; in any untrusted context, data must be encrypted. Satisfying these goals has different levels of complexity depending on the type of cloud service. There are several solutions ensuring confidentiality for the storage as a service paradigm, while guaranteeing confidentiality in the database as a service (DBaaS) paradigm is still an open research area. In this context, we propose SecureDBaaS as the first solution that allows cloud tenants to take full advantage of DBaaS qualities, such as availability, reliability, and elastic scalability, without exposing unencrypted data to the cloud provider.

The architecture design was motivated by a threefold goal: to allow multiple, independent, and geographicallydistributed clients to execute concurrent operations on encrypted data, including SQL statements that modify thedatabase structure; to preserve data confidentiality and consistency at the client and cloud level; to eliminate any intermediate server between the cloud client and the cloud provider. The possibility of combining availability, elasticity,and scalability of a typical cloud DBaaS with data confidentiality is demonstrated through a prototype of Secure DBaaS that supports the execution of concurrent and independent

operations to the remote encrypted database from many geographically distributed clients as in any unencrypted DBaaS setup. To achieve these goals, SecureDBaaS integrates existing cryptographic schemes, isolation mechanisms, and novel strategies for management of encrypted metadata on the untrusted cloud database. This paper contains a theoretical discussion about solutions for data consistency issues due to concurrent and independent client accesses to encrypted data. In this context, we cannot apply fully homomorphic encryption schemes because of their excessive computational complexity. The SecureDBaaS architecture is tailored to cloudplatforms and does not introduce any intermediary proxy or broker server between the client and the cloudprovider. Eliminating any trusted intermediate server allows SecureDBaaS to achieve the same availability,reliability, and elasticity levels of a cloud DBaaS. Other proposals based on intermediateserver(s) were considered impracticable for a cloud-based solution because any proxy represents a single point offailure and a system bottleneck that limits the main benefits (e.g., scalability, availability, and elasticity) of adatabase service deployed on a cloud platform. Unlike SecureDBaaS, architectures relying on a trusted intermediateproxy do not support the most typical cloud scenario where geographically dispersed clients can concurrentlyissue read/write operations and data structure modifications to a cloud database.

## II. LITERATURE SURVEY

This survey intends an architecture that is different from previous work in the field of secure cloud database services. SecureDBaaS (Secure database as a service) architecturesupportsmultiple clients and clients which are geographicallydistributed to execute the independent and concurrentoperation on encrypted data in the remote database.

There are different approaches which guarantees confidentiality [6], [7] by distributing data among different providers and by taking advantage of secret sharing [8]. In such a way, they prevent one cloud provider to read its portion of data, but information can be reconstructed by colluding cloud providers.

A step forward is proposed in [9], that makes it possible to execute range queries on data and to be robust against collusive providers. SecureDBaaS differs from these solutions as it Secure DBaaSdiffers from these solutions as it does not require the use ofmultiple cloud providers, and makes use of SQL-awareencryption algorithms to support the execution of mostcommon SQL operations on encrypted data.

Some DBMS engines offer the possibility of encryptingdata at the filesystem level through the so-called TransparentData Encryption feature [10], [11]. This feature makes itpossible to build a trusted DBMS over untrusted storage.However, the DBMS is trusted and decrypts data beforetheir use. Hence, this approach is not applicable to theDBaaS context considered by SecureDBaas, because weassume that the cloud provider is untrusted.

Other solutions, such as [12], allow the execution ofoperations over encrypted data. These approaches preservedata confidentiality in scenarios where the DBMS is nottrusted; however, they require a modified DBMS engineand are not compatible with DBMS software (bothcommercial and open source) used by cloud providers.On the other hand, SecureDBaaS is compatible withstandard DBMS engines, and allows tenants to build securecloud databases by leveraging cloud DBaaS services already available.

For this reason, SecureDBaaS is more related to[3] and [2] that preserve data confidentiality in untrustedDBMSs through encryption techniques, allow the executionof SQL operations over encrypted data, and are compatiblewith common DBMS engines. However, the architecture ofthese solutions is based on an intermediate and trustedproxy that mediates any interaction between each client andthe untrusted DBMS server.

The approach proposed in [3]by the authors of the DBaaS model [1] works by encryptingblocks of data instead of each data item. Whenever a dataitem that belongs to a block is required, the trusted proxyneeds to retrieve the whole block, to decrypt it, and to filterout unnecessary data that belong to the same block. As aconsequence, this design choice requires heavy modificationsof the original SQL operations produced by eachclient, thus causing significant overheads on both the DBMSserver and the trusted proxy.

Other works [4], [5]introduce optimization and generalization that extend thesubset of SQL operators supported by [3], but they share thesame proxy-based architecture and its intrinsic issues. Onthe other hand, SecureDBaaS allows the execution ofoperations over encrypted data through SQL-aware encryptionalgorithms. This technique, initially proposed inCryptDB [2], makes it possible to execute operations overencrypted data that are similar to operations over plaintextdata. In many cases, the query plan executed by the DBMSfor encrypted and plaintext data is the same.

The reliance on a trusted proxy that characterize [3] and[2] facilitates the implementation of a secure DBaaS, and isapplicable to multitier web applications, which are theirmain focus. However, it causes several drawbacks. Sincethe proxy is trusted, its functions cannot be outsourced toan untrusted cloud provider. Hence, the proxy is meant tobe implemented and managed by the cloud tenant.Availability, scalability, and elasticity of the whole secureDBaaS service are then bounded by availability, scalability,and elasticity of the trusted proxy that becomes a singlepoint of failure and a system bottleneck. Since highavailability, scalability, and elasticity are among theforemost reasons that lead to the adoption of cloudservices, this limitation hinders the applicability of [3]and [2] to the cloud database scenario. SecureDBaaS solvesthis problem by letting clients connect directly to the cloudDBaaS, without the need of any intermediate componentand without introducing new bottlenecks and singlepoints of failure.A proxy-based architecture requiring that any clientoperation should pass through one intermediate server isnot suitable to cloud-based scenarios, in which multipleclients, typically distributed among different locations, needconcurrent access to data stored in the same DBMS. On theother hand, SecureDBaaS supports distributed clientsissuing independent and concurrent SQL operations to thesame database and possibly to the same data.

SecureDBaaSextends our preliminary studies [13] showing that dataconsistency can be guaranteed for some operations byleveraging concurrency isolation mechanisms implementedin DBMS engines, and identifying the minimum isolationlevel required for those statements.

Moreover, we nowconsider theoretically and experimentally a complete set ofSQL operations represented by the TPC-C standard benchmark[14], in addition to multiple clients and differentclient-cloud network latencies that were never evaluated inthe literature.

International Journal of Advance Research in Science and Engineering
Vol. No.5, Issue No. 02, February 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

## III. METHODOGOLY

### 3.1 Architecture Design

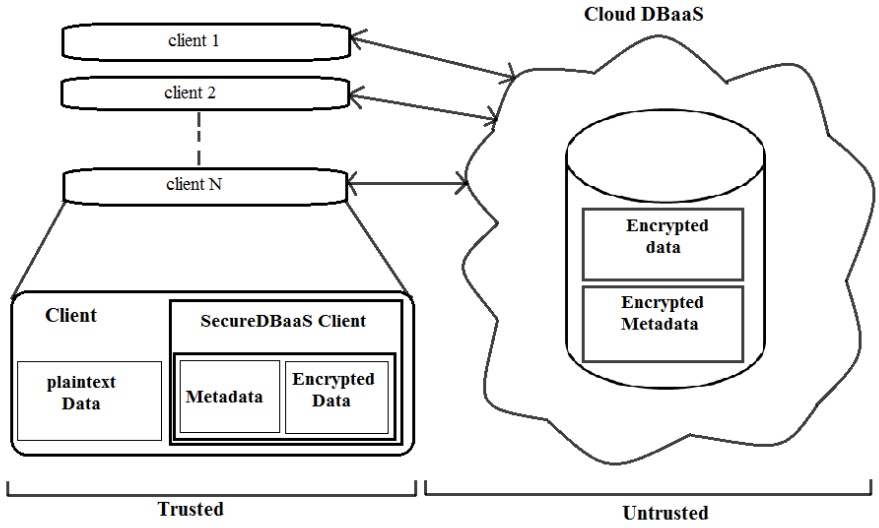


**Figure 1: SecureDBaaS architecture.**

SecureDBaaS is designed to allow multiple and independentclients to connect directly to the untrusted cloudDBaaS without any intermediate server. Figure 1 describes theoverall architecture. We assume that a tenant organizationacquires a cloud database service from an untrusted DBaaSprovider. The tenant then deploys one or more machines(Client 1 through N) and installs a SecureDBaaS client oneach of them. This client allows a user to connect to thecloud DBaaS to administer it, to read and write data, andeven to create and modify the database tables after creation.

SecureDBaaS proposes a different approach where alldata and metadata are stored in the cloud database.SecureDBaaS clients can retrieve the necessary metadatafrom the untrusted database through SQL statements, sothat multiple instances of the SecureDBaaS client can accessto the untrusted cloud database independently with theguarantee of the same availability and scalability propertiesof typical cloud DBaaS. Encryption strategies for tenantdata and innovative solutions for metadata managementand storage are described in the following two sections.

### 3.2 Data Management

We assume that tenant data are saved in a relationaldatabase. We have to preserve the confidentiality of thestored data and even of the database structure because tableand column names may yield information about saved data.We distinguish the strategies for encrypting the databasestructures and the tenant data.

The data type represents the type of the plaintext data(e.g., int, varchar). The encryption type identifies theencryption algorithm that is used to cipher all the data of

a column. It is chosen among the algorithms supported bythe SecureDBaaS implementation. As in [8], SecureDBaaSleverages several SQL-aware encryption algorithms thatallow the execution of statements over encrypted data. It isimportant to observe that each algorithm supports only asubset of SQL operators.When SecureDBaaS creates an encrypted table, the data typeof each column of the encrypted table is determined by

theencryption algorithm used to encode tenant data. Twoencryption algorithms are defined compatible if they produceencrypted data that require the same column data type.

The field confidentiality parameter allows a tenant todefine explicitly which columns of which secure tableshould share the same encryption key (if any). SecureDBaaSoffers three field confidentiality attributes:

- **Column (COL)** is the default confidentiality level that should be used when SQL statements operate on one column; the values of this column are encrypted through a randomly generated encryption key that is not used by any other column.
- **Multicolumn (MCOL)** should be used for columns referenced by join operators, foreign keys, and other operations involving two columns; the two columns are encrypted through the same key.
- **Database (DBC)** is recommended when operations involve multiple columns; in this instance, it is convenient to use the special encryption key that is generated and implicitly shared among all the columns of the database characterized by the same secure type.

### 3.3 Metadata Management

Metadata generated by SecureDBaaS contain all theinformation that is necessary to manage SQL statementsover the encrypted database in a way transparent to theuser. Metadata management strategies represent an originalidea because SecureDBaaS is the first architecture storing allmetadata in the untrusted cloud database together with theencrypted tenant data. SecureDBaaS uses two types ofmetadata.

- Database metadata are related to the whole database. There is only one instance of this metadata type for each database.
- Table metadata are associated with one secure table. Each table metadata contains all information that is necessary to encrypt and decrypt data of the associated secure table.

Database metadata contain the encryption keys that areused for the secure types having the field confidentiality setto database..The structure of a table metadata is represented in Figure 2.Table metadata contain the name of the related secure tableand the unencrypted name of the related plaintext table.

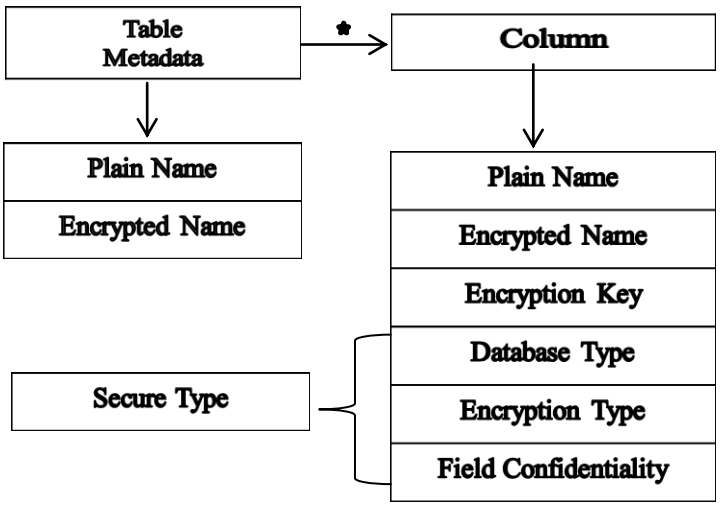


**Figure 2 : Structure of table metadata**

Moreover, table metadata include column metadata for eachcolumn of the related secure table. Each column metadatacontain the following information.

- Plain name: the name of the corresponding column of the plaintext table.
- Coded name: the name of the column of the secure table. This is the only information that links a column to the corresponding plaintext column because column names of secure tables are randomly generated.
- Secure type: the secure type of the column, as defined in Section 3.1. This allows a SecureDBaaS client to be informed about the data type and the encryption policies associated with a column.
- Encryption key: the key used to encrypt and decrypt all the data stored in the column.

This isan original choice that augments flexibility, but opens twonovel issues in terms of efficient data retrieval and dataconfidentiality. To allow SecureDBaaS clients to manipulate metadata through SQL statements, we save database andtable metadata in a tabular form. Even metadata confidentialityis guaranteed through encryption. The structure of themetadata storage table is shown in Figure 3. This table usesone row for the database metadata, and one row for eachtable metadata.

Metadata Storage Table

| ID | Encrypted Metadata | Control Structure |
|---|---|---|
| MAC('.' + Db) | Enc(Db metadata) | MAC(Db metadata) |
| MAC(T1) | Enc(T1 metadata) | MAC(T1 metadata) |
| MAC(T2) | Enc(T2 metadata) | MAC(T2 metadata) |
| | | |

**Figure 3.Organization of database metadata and table metadata in the metadata storage table.**

This mechanism has the further benefit of allowingclients to access each metadata independently, which is animportant feature in concurrent environments. In addition,
SecureDBaaS clients can use caching policies to reduce thebandwidth overhead.

## IV. CONCLUSION

The survey describes an architecture that guarantees confidentiality of data stored in public cloud databases.Unlike state-of-the-art approaches, our solution does not rely on an intermediate proxy that we consider a single point of failure and a bottleneck limiting availability and scalability of typical cloud database services. A large part of the research includes solutions to support concurrent SQL operations (including statements modifying the database structure) on encrypted data issued by heterogeneous and possibly geographically dispersed clients. The architecture does not require modifications to the cloud database, and it is immediately applicable to existing cloud DBaaS, There are no theoretical and practical limits to extend our solution to other platforms and to include new encryption algorithms. In particular, concurrent read and write operations that do not modify the structure of the encrypted database cause negligible overhead.Dynamic scenarios characterized by (possibly) concurrent modifications of the database structure are supported, but at the

price of high computational costs. These performance results open the space to future improvements that we are investigating.

## REFERENCES

[1] H. Hacigu¨mu¨ s,, B. Iyer, and S. Mehrotra, "Providing Database as a Service," Proc. 18th IEEE Int'l Conf. Data Eng., Feb. 2002.

[2] R.A. Popa, C.M.S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: Protecting Confidentiality with Encrypted Query Processing," Proc. 23rd ACM Symp. Operating Systems Principles,Oct. 2011.

[3] H. Hacigu¨mu¨ s,, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," Proc. ACM SIGMOD Int'l Conf. Management Data, June 2002.

[4] J. Li and E. Omiecinski, "Efficiency and Security Trade-Off in Supporting Range Queries on Encrypted Databases," Proc. 19th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security,Aug. 2005.

[5] E. Mykletun and G. Tsudik, "Aggregation Queries in the Database-as-a-Service Model," Proc. 20th Ann. IFIP WG 11.3 Working Conf. Data and Applications Security, July/Aug. 2006.

[6] D. Agrawal, A.E. Abbadi, F. Emekci, and A. Metwally, "DatabaseManagement as a Service: Challenges and Opportunities," Proc.25th IEEE Int'l Conf. Data Eng., Mar.-Apr. 2009.

[7] V. Ganapathy, D. Thomas, T. Feder, H. Garcia-Molina, and R.Motwani, "Distributing Data for Secure Database Services," Proc.Fourth ACM Int'l Workshop Privacy and Anonymity in the InformationSoc., Mar. 2011.

[8] A. Shamir, "How to Share a Secret," Comm. of the ACM,vol. 22, no. 11, pp. 612-613, 1979.

[9] M. Hadavi, E. Damiani, R. Jalili, S. Cimato, and Z. Ganjei, "AS5: ASecure Searchable Secret Sharing Scheme for Privacy PreservingDatabase Outsourcing," Proc. Fifth Int'l Workshop Autonomous andSpontaneous Security, Sept. 2013.

[10]"OracleAdvancedSecurity,"OracleCorporation,http://www.oracle.com/technetwork/database/options/advanced-security,Apr. 2013.

[11] G. Cattaneo, L. Catuogno, A.D. Sorbo, and P. Persiano, "TheDesign and Implementation of a Transparent Cryptographic FileSystem For Unix," Proc. FREENIX Track: 2001 USENIX Ann.Technical Conf., Apr. 2001.

[12] E. Damiani, S.D.C. Vimercati, S. Jajodia, S. Paraboschi, and P.Samarati, "Balancing Confidentiality and Efficiency in UntrustedRelational Dbmss," Proc. Tenth ACM Conf. Computer and Comm.Security, Oct. 2003.

[13] L. Ferretti, M. Colajanni, and M. Marchetti, "Supporting Securityand Consistency for Cloud Database," Proc. Fourth Int'l Symp.Cyberspace Safety and Security, Dec. 2012.

[14] "Transaction Processing Performance Council," TPC-C, http://www.tpc.org, Apr. 2013.