ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Distributed and Concurrent Technique to Access Cloud Databases

K.Siva Prasad
PG Scholar,
Dept of CSE,
St.Mark Educational Institution Society Group of Institutions,
AP, India.

M.Venkatesh Naik
Assistant Professor,
Dept of CSE,
St.Mark Educational Institution Society Group of Institutions,
AP, India.

Ms.M.Shiva lakshmi
Assistant Professor,
Dept of CSE,
St.Mark Educational Institution Society Group of Institutions,
AP, India.

Abstract:
Placing critical data in the hands of a cloud provider should come with the guarantee of security and availability for data at rest, in motion, and in use. Several alternatives exist for storage services, while data confidentiality solutions for the database as a service paradigm are still immature. We propose a novel architecture that integrates cloud database services with data confidentiality and the possibility of executing concurrent operations on encrypted data. This is the first solution supporting geographically distributed clients to connect directly to an encrypted cloud database, and to execute concurrent and independent operations including those modifying the database structure. The proposed architecture has the further advantage of eliminating intermediate proxies that limit the elasticity, availability, and scalability properties that are intrinsic in cloud-based solutions. The efficacy of the proposed architecture is evaluated through theoretical analyses and extensive experimental results based on a prototype implementation subject to the TPC-C standard benchmark for different numbers of clients and network latencies.

1. INTRODUCTION:

In a cloud context, where critical information is placed in infrastructures of untrusted third parties, ensuring data confidentiality is of paramount importance [1], [2]. This requirement imposes clear data management choices: original plain data must be accessible only by trusted parties that do not include cloud providers, intermediaries, 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 (e.g., [3]–[5]), while guaranteeing confidentiality in the database as a service (DBaaS) paradigm [6] is still an open research area. In this context, we propose Secure DBaaS as the first solution that allows cloud tenants to take full advantage of DBaaS qualities, such as availability, reliability, elastic scalability, without exposing unencrypted data to the cloud provider. The architecture design was motivated by a threefold goal: to allow multiple, independent, and geographically distributed clients to execute concurrent operations on encrypted data, including SQL statements that modify the database 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 are demonstrated through a prototype of SecureDBaaS 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 [7] because of their excessive computational complexity. The SecureDBaaS architecture is tailored to cloud platforms and does not introduce any intermediary proxy or broker server between the client and the cloud provider. Eliminating any trusted intermediate server allows SecureDBaaS to achieve the same availability, reliability and elasticity levels of a cloud DBaaS. Other proposals (e.g., [8]–[11]) based on intermediate server(s) were considered impracticable for a cloud-based solution because any proxy represents a single point of failure and a system bottleneck that limits the main benefits (e.g., scalability, availability, elasticity) of a database service deployed on a cloud platform. Unlike SecureDBaaS, architectures relying on a trusted intermediate proxy do not support the most typical cloud scenario where geographically dispersed clients can concurrently issue read/write operations and data structure modifications to a cloud database.

A large set of experiments based on real cloud platforms demonstrate that Secure DBaaS is immediately applicable to any DBMS because it requires no modification to the cloud database services. Other studies where the proposed architecture is subject to the TPC-C standard benchmark for different numbers of clients and network latencies show that the performance of concurrent read and write operations not modifying the Secure DBaaS database structure is comparable to that of unencrypted cloud database. Workloads including modifications to the database structure are also supported by Secure DBaaS, but at the price of overheads that seem acceptable to achieve the desired level of data confidentiality. The motivation of these results is that network latencies, which are typical of cloud scenarios, tend to mask the performance costs of data encryption on response time. The overall conclusions of this paper are important because for the first time they demonstrate the applicability of encryption to cloud database services in terms of feasibility and performance.

The remaining part of this paper is structured as following. Section 2 compares our proposal to existing solutions related to confidentiality in cloud database services. Section 3 and Section 4 describe the overall architecture and how it supports its main operations, respectively. Section 5 reports some experimental evaluation achieved through the implemented prototype. Section 6 outlines the main results. Space limitation requires us to postpone the assumed security model in Appendix A, to describe our solutions to concurrency and data consistency problems in Appendix B, to detail the prototype architecture in Appendix C.

2 RELATED WORK:

SecureDBaaS provides several original features, that differentiate it from previous work in the field of security for remote database services.

• It guarantees data confidentiality by allowing a cloud database server to execute concurrent SQL operations (not only read/write, but also modifications to the database structure) over encrypted data.

• It provides the same availability, elasticity, and scalability of the original cloud DBaaS because it does not require any intermediate server. Response times are affected by cryptographic overheads that for most SQL operations are masked by network latencies

. • Multiple clients, possibly geographically distributed, can access concurrently and independently to a cloud database service.

• It does not require a trusted broker or a trusted proxy because tenant data and metadata stored by the cloud database are always encrypted.

• It is compatible with the most popular relational database servers, and it is applicable to different DBMS implementations because all adopted solutions are database agnostic.

Cryptographic file systems and secure storage solutions represent the earliest works in this field. We do not detail the several papers and products (e.g., Sporc [3], Sundr [4], Depot [5]) because they do not support computations on encrypted data.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Different approaches guarantee some confidentiality (e.g., [12], [13]) by distributing data among different providers and by taking advantage of secret sharing [14]. 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 [15], that makes it possible to execute range queries on data and to be robust against collusive providers. SecureDBaaS differs from these solutions as it does not require the use of multiple cloud providers, and makes use of SQL-aware encryption algorithms to support the execution of most common SQL operations on encrypted data. SecureDBaaS relates more closely to works using encryption to protect data managed by untrusted databases. In such case, a main issue to address is that cryptographic techniques cannot be na¨ıvely applied to standard DBaaS because DBMS can only execute SQL operations over plaintext data. Some DBMS engines offer the possibility of encrypting data at the filesystem level through the so called Transparent Data Encryption feature [16], [17].

This feature makes it possible to build a trusted DBMS over untrusted storage. However, the DBMS is trusted and decrypts data before their use. Hence, this approach is not applicable to the DBaaS context considered by SecureDBaas, because we assume that the cloud provider is untrusted. Other solutions, such as [18], allow the execution of operations over encrypted data. These approaches preserve data confidentiality in scenarios where the DBMS is not trusted, however they require a modi- fied DBMS engine and are not compatible with DBMS software (both commercial and open source) used by cloud providers. On the other hand, SecureDBaaS is compatible with standard DBMS engines, and allows tenants to build secure cloud databases by leveraging cloud DBaaS services already available. For this reason, SecureDBaaS is more related to [9] and [8] that preserve data confidentiality in untrusted DBMSs through encryption techniques, allow the execution of SQL operations over encrypted data, and are compatible with common DBMS engines.

However, the architecture of these solutions is based on an intermediate and trusted proxy that mediates any interaction between each client and the untrusted DBMS server. The approach proposed in [9] by the same authors of the DBaaS model [6], works by encrypting blocks of data instead of each data item. Whenever a data item that belongs to a block is required, the trusted proxy needs to retrieve the whole block, to decrypt it, and to filter out unnecessary data that belong to the same block. As a consequence, this design choice requires heavy modifications of the original SQL operations produced by each client, thus causing significant overheads on both the DBMS server and the trusted proxy. Other works [10], [11] introduce optimization and generalization that extend the subset of SQL operators supported by [9], but they share the same proxy-based architecture and its intrinsic issues. On the other hand, SecureDBaaS allows the execution of operations over encrypted data through SQL-aware encryption algorithms. This technique, initially proposed in CryptDB [8], makes it possible to execute operations over encrypted data that are similar to operations over plaintext data. In many cases, the query plan executed by the DBMS for encrypted and plaintext data is the same.

3 ARCHITECTURE DESIGN:

SecureDBaaS is designed to allow multiple and independent clients to connect directly to the untrusted cloud DBaaS without any intermediate server. Figure 1 describes the overall architecture. We assume that a tenant organization acquires a cloud database service from an untrusted DBaaS provider. The tenant then deploys one or more machines (Client 1 through N) and install a SecureDBaaS client on each of them. This client allows a user to connect to the cloud DBaaS to administer it, to read and write data, and even to create and modify the database tables after creation. We assume the same security model that is commonly adopted by the literature in this field (e.g., [8], [9]), where: tenant users are trusted, the network is untrusted, and the cloud provider is honest-but-curious, that is, cloud service operations are executed correctly, but tenant information confidentiality is at

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

risk. For these reasons, tenant data, data structures, and metadata must be encrypted before exiting from the client. A thorough presentation of the security model adopted in this paper is in Appendix A. The information managed by SecureDBaaS includes plaintext data, encrypted data, metadata, and encrypted metadata. Plaintext data consist of information that a tenant wants to store and process remotely in the cloud DBaaS. To prevent an untrusted cloud provider from violating confidentiality of tenant data stored in plain form, SecureDBaaS adopts multiple cryptographic techniques to transform plaintext data into encrypted tenant data, and encrypted tenant data structures because even the names of the tables and of their columns must be encrypted. SecureDBaaS clients produce also a set of metadata consisting of information required to encrypt and decrypt data as well as other administration information. Even metadata are encrypted and stored in the cloud DBaaS.

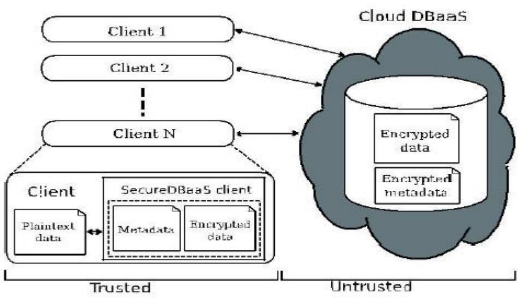

Fig. 1. SecureDBaaS architecture.

SecureDBaaS moves away from existing architectures that store just tenant data in the cloud database, and save metadata in the client machine [9] or split metadata between the cloud database and a trusted proxy [8]. When considering scenarios where multiple clients can access the same database concurrently, these previous solutions are quite inefficient. For example, saving metadata on the clients would require onerous mechanisms for metadata synchronization, and the practical impossibility of allowing multiple clients to access cloud database services independently. Solutions based on a trusted proxy are more feasible, but they introduce a system bottleneck that reduces availability, elasticity and scalability of cloud database services.

3.1 Data management:

We assume that tenant data are saved in a relational database. We have to preserve the confidentiality of the stored data and even of the database structure because table and column names may yield information about saved data. We distinguish the strategies for encrypting the database structures and the tenant data. Encrypted tenant data are stored through secure tables into the cloud database. To allow transparent execution of SQL statements, each plaintext table is transformed into a secure table because the cloud database is untrusted. The name of a secure table is generated by encrypting the name of the corresponding plaintext table. Table names are encrypted by means of the same encryption algorithm and an encryption key that is known to all the SecureDBaaS clients. Hence, the encrypted name can be computed from the plaintext name. On the other hand, column names of secure tables are randomly generated by SecureDBaaS, hence even if different plaintext tables have columns with the same name, the names of the columns of the corresponding secure tables are different.

This design choice improves confidentiality by preventing an adversarial cloud database from guessing relations among different secure tables through the identification of columns having the same encrypted name. SecureDBaaS allows tenants to leverage the computational power of untrusted cloud databases by making it possible to execute SQL statements remotely and over encrypted tenant data, although remote processing of encrypted data is possible to the extent allowed by the encryption policy. To this purpose, SecureDBaaS extends the concept of data type, that is associated to each column of a traditional database by introducing the secure type. By choosing a secure type for each column of a secure table, a tenant can define fine-grained encryption policies, thus reaching the desired trade-off between data confidentiality and remote processing ability. A secure type is composed by three fields: data type, encryption type, and field confidentiality.

The combination of the encryption type and of the field confidentiality parameters defines the encryption policy of the associated column.

3.2 Metadata management:

Metadata generated by SecureDBaaS contain all the information that is necessary to manage SQL statements over the encrypted database in a way transparent to the user. Metadata management strategies represent an original idea because SecureDBaaS is the first architecture storing all metadata in the untrusted cloud database together with the encrypted tenant data. SecureDBaaS uses two types of metadata.

• 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.

This design choice makes it possible to identify which metadata type is required to execute any SQL statement so that a SecureDBaaS client needs to fetch only the metadata related to the secure table/s that is/are involved in the SQL statement. Retrieval and management of database metadata are necessary only if the SQL statement involves columns having the field confidentiality policy equal to database. This design choice minimizes the amount of metadata that each SecureDBaaS client has to fetch from the untrusted cloud database, thus reducing bandwidth consumption and processing time. Moreover, it allows multiple clients to access independently metadata related to different secure tables, as we discuss in Section 4.3 and Appendix B. Database metadata contain the encryption keys that are used for the secure types having the field confi- dentiality set to database. A different encryption key is associated with all the possible combinations of data type and encryption type. Hence, the database metadata represent a keyring and do not contain any information about tenant data.

4 OPERATIONS:

In this section we outline the setup setting operations carried out by a database administrator (DBA), and we describe the execution of SQL operations on encrypted data in two scenarios: a na¨ıve context characterized by a single client, and realistic contexts where the database services are accessed by concurrent clients.

4.1 Setup phase:

We describe how to initialize a SecureDBaaS architecture from a cloud database service acquired by a tenant from a cloud provider. We assume that the DBA creates the metadata storage table that at the beginning contains just the database metadata, and not the table metadata. The DBA populates the database metadata through the SecureDBaaS client by using randomly generated encryption keys for any combinations of data types and encryption types, and stores them in the metadata storage table after encryption through the master key. Then, the DBA distributes the master key to the legitimate users. User access control policies are administrated by the DBA through some standard data control language as in any unencrypted database.

In the following steps, the DBA creates the tables of the encrypted database. He must consider the three field confidentiality attributes (COL, MCOL, DBC) introduced at the end of the Section 3. Let us describe this phase by referring to a simple but representative example shown in Figure 4, where we have three secure tables named ST1, ST2 and ST3. Each table STi (i = 1, 2, 3) includes an encrypted table Ti that contains encrypted tenant data, and a table metadata Mi. (Although in the reality the names of the columns of the secure tables are randomly generated, for the sake of simplicity, this figure refers to them through C1-CN.)

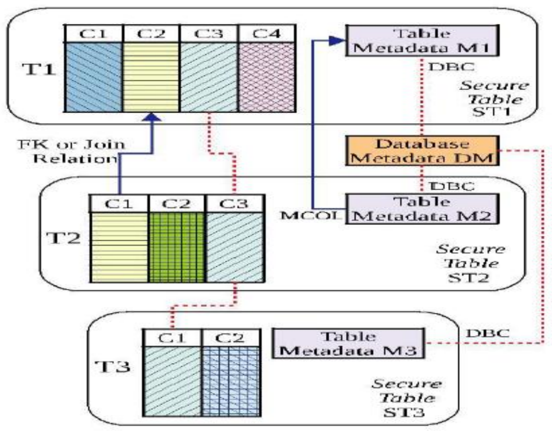

Fig. 4. Management of the encryption keys according to the field confidentiality parameter.

4.2 Sequential SQL operations:

We describe the SQL operations in SecureDBaaS by considering an initial simple scenario in which we assume that the cloud database is accessed by one client. Our goal here is to highlight the main processing steps, hence we do not take into account performance optimizations and concurrency issues that will be discussed in Section 4.3 and Appendix B. The first connection of the client with the cloud DBaaS is for authentication purposes: SecureDBaaS relies on standard authentication and authorization mechanisms provided by the original DBMS server. After the authentication, a user interacts with the cloud database through the SecureDBaaS client. SecureDBaaS analyzes the original operation to identify which tables are involved and to retrieve their metadata from the cloud database. The metadata are decrypted through the master key and their information is used to translate the original plain SQL into a query that operates on the encrypted database.

5 CONCLUSIONS:

We propose an innovative 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 heterogenous and possibly geographically dispersed clients. The proposed architecture does not require modifications to the cloud database, and it is immediately applicable to existing cloud DBaaS, such as the experimented PostgreSQL Plus Cloud Database [23], Windows Azure [24] and Xeround [22]. There are no theoretical and practical limits to extend our solution to other platforms and to include new encryption algorithms. It is worth to observe that experimental results based on the TPC-C standard benchmark show that the performance impact of data encryption on response time becomes negligible because it is masked by network latencies that are typical of cloud scenarios. 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] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica et al., “A view of cloud computing,” Communications of the ACM, vol. 53, no. 4, 2010.

[2] W. Jansen and T. Grance, “Guidelines on security and privacy in public cloud computing,” Tech. Rep. NIST Special Publication 800-144, 2011.

[3] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten, “Sporc: group collaboration using untrusted cloud resources,” in Proc. of the 9th USENIX conference on Operating Systems Design and Implementation, October 2010.

[4] J. Li, M. Krohn, D. Mazieres, and D. Shasha, “Secure untrusted ` data repository (sundr),” in Proc.

of the 6th USENIX conference on Opearting Systems Design and Implementation, October 2004.

[5] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, “Depot: Cloud storage with minimal trust,” ACM Transactions on Computer Systems, vol. 29, no. 4, 2011.

[6] H. Hacigum¨ us¨ ,, B. Iyer, and S. Mehrotra, “Providing database as a service,” in Proc. of the 18th IEEE International Conference on Data Engineering, February 2002.