TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol. 16, No. 2, November 2015, pp. 333 ~ 340
DOI: 10.11591/telkomnika.v16i2.8430

SecureDBaaS Model for Accessing Encrypted Cloud Databases

P. Jagadeeswaraiah, M.R. Pavan Kumar
Sree Vidyanikethan Engineering College (SVEC), Tirupati, AP, India
Corresponding author, e-mail: jagadesh.palle@gmail.com, sivapavan.mr@gmail.com

Abstract

Cloud computing has recently emerged being a compelling paradigm that pertains to managing and delivering services over the web. The particular prevalent problem connected with cloud is confidentiality, security, as well as reliability etc., in which how the cloud provider assures. To recognize this, a novel architecture is usually introduced that will integrates cloud database services and as well executing concurrent operations on encrypted information. Also a new homomorphic encryption algorithm will likely be incorporated to offer confidentiality as well as concurrent execution of various SQL operations. This will be the first option supporting quite a few distributed clienteles to access encrypted cloud databases. One of main thing is that it eliminates advanced proxies in between cloud user and provider.The performance on the architecture is usually calculated by means of theoretical and practical results which are subjected to TPC-C benchmark standard tools for a number of clients as well as network latencies.

Keyword: *cloud, security, homomorphic encryption, confidentiality*

1. Introduction

In a cloud era, critical data is placed in infrastructure of third parties services assuring the protection and confidentiality may be the prior importance. By which the cloud provides as well as the third parties services has the availability of accessing the confidential information from the clients. So the original plain data is available by the trusted parties as well as remaining untrusted context data should be encrypted. There are many solutions ensuring the confidentiality for storage as a service but confidentiality from the dbaas continues to be in naive stage.

The architecture design is motivated because of the multiple and independent clients to perform the operations within the secured data because of the SQL statements, which could modify the database structure. Database as a service is an important and operational technique enabling IT providers to provide database functionality being a service to number of consumers. The propose architecture provides the property of making the independent and parallel operations for the remote encrypted database from any geographically located clients, in any unencrypted database setup. In the proposed system we have been not including any intermediate proxy relating to the client and the particular cloud provider. The Secure database as a service is actually immediately applicable to any DBMS as it doesn't require any modifications for the cloud database. Along with the other part regarding architecture is tailored for the homomorphic encryption algorithmic process that provides confidentiality by implementing various computations on data.

A huge set of tests on real cloud platforms elucidates that SecureDBaaS is straight away applicable to any package as a result of it needs no modification on the cloud data services. Other studies where ever the proposed design is subjected to the TPC-C standard benchmark tools for various clients and network latencies shows which the performance of synchronic read and write procedures does not modify SecureDBaaS database structure. Thus suggested architecture provides security by applying the homomorphic encryption on the information of clients along with the response time with the read/write operations may decrease as for the reason that entire database is usually encrypted. Overall the conclusions of paper are crucial because for the first time demonstrates the applicability of encryption to cloud database services in terms of performance and overheads.

Received June 21, 2015; Revised September 24, 2015; Accepted October 17, 2015

2. Related Work

SecureDBaaS gives several unique features that differentiate it from earlier work in the field of security intended for remote databases services.

a) It assures files secrecy by enabling some sort of cloud database server to execute concurrency SQL operations (not only read/write) over encrypted files.
b) It provides the availability, elasticity, and scalability of the original cloud dbaas because it does not require any proxy server.
c) Response times are affected by cryptographic overheads for most of SQL operations are masked by network latencies.
d) It does not involve a reliable proxy because tenant data and metadata saved by the cloud database are usually encrypted.

Cryptographic systems and secure storage options represent the earliest works in this field. We don't depth the many documents and products (e.g., Sporc, Sundr, and Depot) because they cannot support computations on protected data.

L. Ferretti proposed an architecture that reduces the risk for any intermediary part, thus achieving availability and scalability just like that of unencrypted cloud database services. Advantages are Guarantees file consistency in scenarios during which independent clients simultaneously execute SQL requests, and the structure from the database can end up being modified. Reduced isolation quantities for multi-version systems have not been characterized before despite being implemented in several products and the drawbacks are concurrent modifications from the database structure are generally supported but at the expense of higher overhead as well as stricter transaction remote location levels. In that paper, we present CryptDB because intermediate server involving the client and server delivers confidentiality for request that uses DBMSs. CryptDB's approach should be to execute queries in excess of encrypted data as well as the key that SQL runs on the well-defined set while operators. The benefits are, CryptDB stops the DBA through learning private files. CryptDB ensures your confidentiality of logged out users data. Their drawbacks are generally Throughput penalty arises while tracing your database using MYSQL servers is looks like its modest or unassembled. This paper overcomes the issue between a client as well as the server while processing the clients issue request. In that paper we advise, using multiple service providers so that you can store data. This process may use the decomposition algorithm that the columns of a database can be split across your server. This algorithm should match the following:

a) Privacy constraints should not be violated
b) Workload should be reduced

Various approaches guarantee that database as a service provides clients seamless mechanisms to produce, store, and access their databases in the host site. NetDB2, a database model on-line provides a useful mechanism for organizations to acquire data management like a service. Database as an email finder service makes the benefit of additional overhead of remote access to data, an infrastructure to guarantee data privacy, and program design. Data privacy can be achieved in different levels by utilizing encryption techniques with both software and hardware levels. NetDB2 model also performs create/remove furniture, views, triggers, crawls, abstract data varieties, SQL queries, create and call user defined functions and stored procedures, producing and deleting crawls, etc. Some dbms engines offer the chance of encrypting knowledge at the file system level through the alleged Transparent Information Security feature. This feature makes it possible to construct a reliable dbms over untrusted storage. The dbms is trusted and decrypts knowledge before their use. This approach is not applicable to the dbms situation regarded by SecureDBaaS. Since we believe that the cloud company is untrusted.

The reliance on the trusted proxy in which it allows for the implementation of secure dbaas, and it is applicable in order to multitier world-wide-web applications, which might be their major focus. This causes various drawbacks. Considering that the proxy is actually trusted, its functions are not outsourced for an untrusted fog up provider. The proxy is supposed to always be implemented as well as managed because of the cloud renter. Availability, scalability as well as elasticity of the whole safeguarded dbaas support are then bounded by means of availability, scalability as well as elasticity of the trusted proxy that becomes 1 point involving failure and also a system bottleneck. Given that high availability, scalability as well as elasticity are one of many foremost factors that lead to the adopting of fog up services, this limitation hinders your applicability for the cloud data source scenario. SecureDBaaS solves this challenge by making

clients connect directly to the DBaaS, without necessity of any kind of intermediate part and without introducing completely new bottlenecks as well as single factors of failing.

A new proxy based buildings requiring which any buyer operation should move through one advanced beginner server isn't suitable to help cloud-based situations, in which in turn multiple clientele, typically sent out among various locations, need concurrent use of data stored from the same DBMS. On the other hand, SecureDBaaS sustains distributed clientele issuing self-sufficient and contingency SQL operations towards same database and perhaps to identical data. SecureDBaaS expands our early studies exhibiting that files consistency can be guaranteed for some operations simply by leveraging concurrency isolation mechanisms carried out in DBMS engines, and discovering the lowest isolation level important for those claims. Moreover, we have now consider in theory and experimentally a total set regarding SQL procedures represented by the TPC-C typical benchmark, besides multiple clients and different client-cloud circle latencies that have been never evaluated from the literature.

3. Motivation

Sustaining critical data in the hands of a cloud provider is often a challenging one that is certainly associated with secrecy. There exist quite a few encryption techniques now available that provides the particular guarantee of safety measures and availability for data. To provide each of the security features to data there were different types of architectures; algorithms are increasingly being developed and are in naive stage. Feasibility, functionality and performance can be calculated using typical tested like emulab and cloud providers like amazon EC2, windows azure, Xeround, Rackspace etc.

4. Problem Definition

Today most of the information is being maintained in the hands of cloud providers. But, how far the data is secure and available whether the data is in use, in motion and at rest. This is a critical phenomenon and there exists several possible solutions too. Also how securely cloud databases can be accessed.

5. Problem Statement

SecureDBaaS is made to allow numerous and separate clients to connect to cloud without intermediate server. Data and also metadata are generally encrypted just before upload towards the cloud. Multiple cryptography techniques are widely-used to transform plain word into encrypted data. Table titles and the column names can also be encrypted inside cloud database using safety scheme. The device supports geographically spread clients to connect directly a great encrypted cloud database. The client can execute concurrent query processing encrypted sources. Homomorphic encryption operations like RSA, ElGamal, Pailler systems are widely-used in the machine. Concurrent data and data structure changes are allowed for the cloud sources. The following problems are generally identified through the existing method.

a) Database structure modification increases the computational overhead
b) Data integrity verification is not performed
c) Access control mechanism is not provided
d) Query submission is not secured

6. Problem Formulation

With cloud research technology, there were plenty of important policy issues may include like privacy, secrecy, anonymity, government surveillance, stability etc., of these entire most crucial one will be security, through which how the cloud provider assures. Generally cloud computing features different class of customers that features academicians, everyday users and also enterprises etc., in which in turn their motivation is always to move about the cloud regarding deploying software, managing assets etc. Security standpoint is different perspective regarding for different consumers. Suppose regarding enterprises high performance may become criteria whereas for academicians it can be on the performance regarding computing. Like this several issues occur in the cloud.

7. Architecture Design

The architecture layout of SecureDBaaS is made up of one or more client devices with SecureDBaaS installed and also cloud database. The Figure 1 gives entire architecture. This certain client is likely for the text of anyone to this particular cloud dbaas to be able to perform SQL operations. To combat the confidentiality troubles, many cryptographic strategies are widely-used to convert plaintext data to encrypted variety for storing inside cloud database. The architecture consists of five types of information. Such as:

Plain data: the informative content provided by the client users.

Encrypted data: the encrypted data that is stored in the cloud database.

Plain metadata: all the information required by the clients to manage encrypted data on the cloud database.

Encrypted metadata: the encrypted metadata that is stored in the cloud database.

Master key: the encryption key of the encrypted metadata. This key is distributed to all the clients.

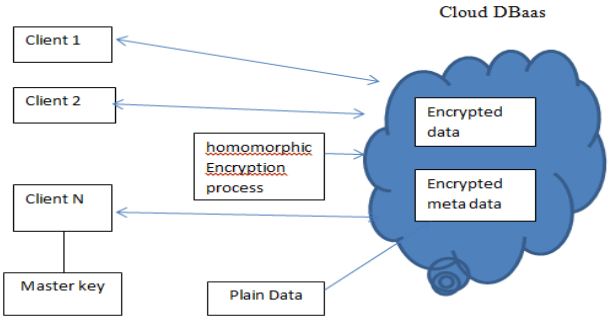

Figure 1. SecureDBaaS Architecture

Here we assume that tenant firm acquires the cloud database service from untrusted dbaas provider. The tenant then deploys a number of machines and installs the SecureDBaaS consumer on each of them. This consumer allows the user to touch this cloud dbaas to manage it, just read and compose data, as well as to build and change the repository tables after creation. We assume the identical security model that is commonly adopted where tenant users usually are trusted, the circle is untrusted plus the cloud provider is honest-but-curious, cloud service operations are accomplished correctly, however tenant info confidentiality threat. The info managed through SecureDBaaS includes plaintext data, encrypted data, metadata, and encrypted metadata. Plaintext data incorporate information that your tenant wants to store and process remotely inside cloud dbaas. To stop an untrusted impair provider by violating confidentiality of tenant data located in simple form, SecureDBaaS switches into multiple cryptographic approaches to transform plaintext data into encrypted tenant data and encrypted tenant data set ups because even the names from the tables and of these columns has to be encrypted. SecureDBaaS consumers produce also a set of metadata composing of information forced to encrypt decrypt data as well as other administration info. Even metadata usually are encrypted and stored inside cloud dbaas. SecureDBaaS moves away from existing architectures which store just tenant data inside cloud repository, and spend less metadata inside client appliance or separated metadata involving the cloud database along with a trusted proxy. When contemplating scenarios exactly where multiple consumers can access the identical database at the same time previous solutions are quite inefficient. Solutions depending on a trustworthy proxy tend to be feasible; nonetheless they introduce a head unit bottleneck which reduces access, elasticity and scalability involving cloud repository services. SecureDBaaS proposes a new approach exactly where all data and metadata usually are stored throughout the cloud database. SecureDBaaS consumers can retrieve the mandatory metadata from the untrusted repository through SQL statements, so which multiple instances of the SecureDBaaS consumer can access to the untrusted impair database independently while using the guarantee from the same access and scalability properties of typical cloud dbaas. Encryption strategies for tenant data and modern solutions intended for metadata administration and safe-keeping are explained.

8. Problem Solving

8.1. Homomorphic Encryption Applied to Cloud Security

When the data transferred in the cloud we employ standard encryption approaches to secure the actual operations along with the storage from the data. Our essential concept seemed to be to encrypt the data before deliver it to the Cloud provider. But a final one should decrypt info at each and every operation. The client will likely need to provide the actual private key to the server (Cloud provider) to decrypt data before performing the calculations required, that may affect the actual confidentiality and privacy regarding data stored from the Cloud. Here we execute operations on protected information without decrypting them, which may provide the exact same results after calculations like we have worked on the raw data. Homomorphic Encryption programs are used to accomplish operations on encrypted data without knowing the private key (without decryption), the customer is the only real owner of the secret key.

Def.: An encryption is homomorphic, if from Enc(x) and Enc(y) it is possible to compute Enc(f (x, y)), where f can be: +, ×, ⊕ and without using the private key .One of the Homomorphic encryption we recognize, according to the operations that enables to evaluate on information, the additive Homomorphic encryption (only additions of the data) could be the Pailler and Goldwasser-Micalli cryptosystems, and the multiplicative Homomorphic encryption (only products on data) could be the RSA and El Gamal cryptosystems.

Suppose Ek is an encryption algorithm with key k.
Dk is a decryption algorithm with key k.
Dk (Ek (p) × Ek (q)) = p×q OR Enc (x⊗y) = Enc(x) ⊗ Enc(y)
Dz (Ez (n) × Ez (m)) = n+m OR Enc (x⊕y) = Enc(x) ⊗ Enc(y)

The first property is called multiplicative homomorphic encryption, and the second is additive homomorphic encryption. An algorithm is fully homomorphic if both properties are satisfied simultaneously.

8.2. Homomorphic Encryption (RSA cryptosystem)

The below given figure 2 gives us homomorphic encryption.

Let $n = pq$ where p and q are primes. Pick a and b such that $ab \equiv 1 \pmod{\phi(n)}$. n and b are public while p, q and a are private.

$$e_K(x) = x^b \bmod n$$

$$d_K(y) = y^a \bmod n$$

The Homomorphism: Suppose p1 and p2 are plaintexts. Then,

$Ek(p1)\ Ek(p2) = (p1)_b(p2)_b$

$\bmod n = (p1p2)_b \bmod n = Ek(p1p2)$

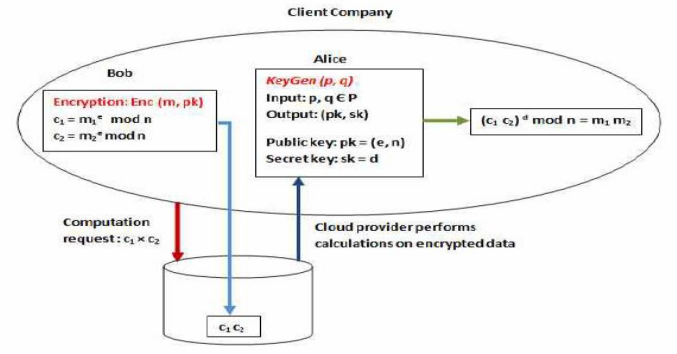

Figure 2. Multiplicative Homomorphic Encryption

8.3. Sequential SQL Operations

We identify the SQL operations in SecureDBaaS simply by considering an initial simple scenario by which we assume that the cloud repository is utilized by a single client. Our goal suggestions to highlight the principle processing measures; we usually do not think about performance optimizations and also concurrency. The first connection on the client with the cloud DBaaS is for authenticated functions. SecureDBaaS utilizes standard authentication and also authorization mechanisms provided by the authentic DBMS server. After the authentication, a individual interacts with the cloud database over the SecureDBaaS purchaser. SecureDBaaS analyzes the original operation to be able to identify that tables could happen and to be able to retrieve their metadata from your cloud repository. The metadata are decrypted over the master key and their particular information is employed to translate the plain SQL into a query which operates for the encrypted repository. Translated operations are then executed with the cloud database above the encrypted data. Because there can be a one-to-one messages between Plaintext and encrypted text, it may be possible to prevent a reliable database individual from being able to view or editing some renter data simply by granting restricted privileges in some furniture. User privileges might be managed directly with the untrusted and also encrypted cloud database. The outcomes of the particular translated dilemma that contains encrypted renter data and also metadata tend to be received with the SecureDBaaS purchaser, decrypted and also delivered towards the user. The complexity on the translation process is dependent upon any type of SQL affirmations depends on the type of SQL statement.

8.4. Concurrent SQL Operations

The support to concurrent delivery of SQL claims issued by numerous separate clients is unquestionably one of the most critical advantages of SecureDBaaS regarding state-of-the-art solutions. Our architecture should guarantee consistency among protected tenant data and protected metadata because broken or out-of-date metadata would prevent clients from decoding protected tenant data producing lasting data losses. A rigorous analysis of the probable problems and answers linked to concurrent SQL procedures on protected tenant data obtainable in the web extra material. Here, we remark the importance of unique two classes of claims that are reinforced by SecureDBaaS: SQL procedures perhaps not producing improvements to the database structure, such as for example for instance read, create, and update; procedures involving variations of the database structure through development, removal and adjustment of database tables. In cases known with a fixed database structure, SecureDBaaS allows clients to concern concurrent SQL instructions to the protected cloud database without presenting any new consistency problems regarding unencrypted databases. After metadata collection, a plaintext SQL order is translated in to one SQL order running on protected tenant data. As metadata don't modify, a consumer may read them when and cache them for further uses, hence increasing performance. SecureDBaaS is the first architecture that allows concurrent and consistent accesses even when you can find procedures that might change the database structure. Such cases, we have to guarantee the consistency of data and metadata through solitude degrees, like the photo solitude that people show may benefit most consumption scenarios.

9. Experimental Results

We display the applicability of SecureDBaaS to various cloud dbaas alternatives by employing and managing protected database procedures on emulated cloud infrastructures. The current edition of SecureDBaaS prototype supports PostgreSQL, MySQL, and SQL Server relational databases. As an initial result, we could discover that porting SecureDBaaS to various DBMS requires modest improvements related to the database connector, and small adjustments of the database.

We carry out a group of tests that assesses the performance and the overheads of our prototype. We utilize the benchmark performance tools like Dell benchmark factory, Monyog, HammerDB that provide us a managed atmosphere with many machines, ensuring repeatability of the tests for the variety of situations to take into account in terms of workload types, quantity of clients, and network latencies. Because the workload model for the repository, we reference the TPC-C standard. The dbms machine is MYSQL 5.6 used on an Intel processor having 8 GB of RAM. Clients are connected to the machine through a LAN, wherever we can present

arbitrary system latencies to emulate WAN connections which can be normal of cloud services. The studies evaluate the expense of encryption, examine the reaction times of simple versus secured repository procedures, and analyze the effect of system latency. We consider two TPC-C certified databases with 10 warehouses that have exactly the same quantity of tuples: simple tuples include 1,046 MB information, while SecureDBaaS tuples have size equal to 1024 MB as a result of encryption overhead. Both databases use repeatable study (snapshot) isolation stage.

To evaluate response times, the customer steps the performance times of the 44 SQL commands to the TPC-C benchmark. Response times are plotted on the histogram of the figure 3 that has a logarithmic Y-axis. TPC-C procedures are grouped on the foundation of the school transaction: Order Status, Stock Level, Payment, and New Order, Delivery. Out of this determine, we are able to enjoy that the response time is less than 3 ms for many procedures, and under 1 ms for nearly all procedures

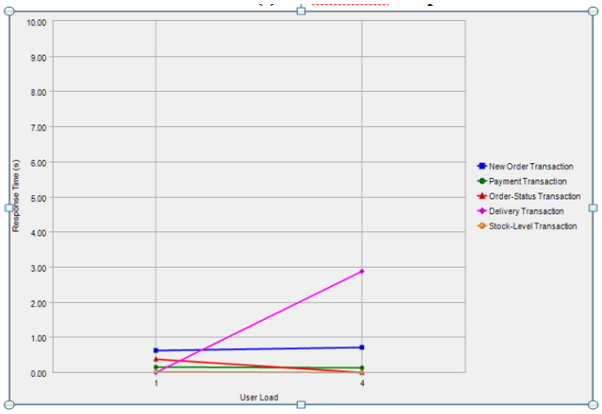

Figure 3. Response times of TPC-C benchmark operations grouped by the transaction class

The next set of experiments is done that gives the throughput for increasing variety of concurrent customers in contexts characterized by 40 ms system latencies, respectively. The below given figure 4 represents TPC-C performance that corresponds to variety of users. Y - axis presents the amount of committed TPC-C transactions each and every minute executed by the clients. The traits of the SecureDBaaS lines are close to those of the initial TPC-C repository, therefore demonstrating that SecureDBaaS protected repository does not influence scalability regarding the simple database. Much more essential, the system latencies tend to mask cryptographic overheads for numerous clients. Like, the overheads of SecureDBaaS with 40 concurrent customers diminish from 20 per cent in a 40-ms situation to 13 percent in a sensible situation. That effect is essential as it confirms that SecureDBaaS is just a legitimate and useful solution for guaranteeing confidentiality in real cloud services.

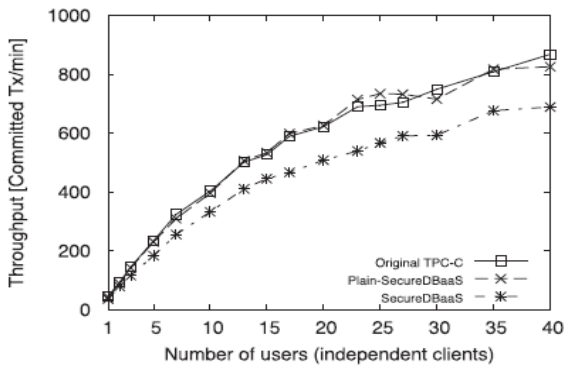

Figure 4. TPC-C Performances

10. Conclusion

We propose an impressive architecture that guarantees confidentiality of data in public cloud databases. Unlike state-of-the-art strategies, our solution doesn't rely on an intermediate proxy that people think about a simple level of disappointment and a bottleneck decreasing supply and scalability of normal cloud database services. A big part of the research contains alternatives to guide concurrent SQL operations (including claims adjusting the database structure) on secured knowledge given by heterogeneous and probably geographically distributed clients. The proposed architecture doesn't need changes to the cloud database, and it's instantly appropriate to current cloud dbaas. We find no theoretical and sensible limits to extend our treatment for other tools and to incorporate new encryption algorithms .Its price observing that experimental benefits based on the TPC-C benchmark standard shows that the performance effect of information encryption on result time becomes minimal because it's negligible by network latencies that are typical of cloud scenarios. Specifically, concurrent study and create procedures that maybe not change the framework of the secured database trigger minimal overhead.

Acknowledgements

I would like to thank M.R. Pavan Kumar for his constructive guidance and encouragement for the paper work.

References

[1] AJ Feldman, WP Zeller, MJ. Freedman, EW Felten. *SPORC: Group Collaboration Using Untrusted Cloud Resources.* Proc. Ninth USENIX Conf. Operating Systems Design and Implementation. 2010.
[2] L Ferretti, M Colajanni, M Marchetti. *Supporting Security and Consistency for Cloud Database.* Proc. Fourth Int'l Symp Cyberspace Safety and Security. 2012.
[3] Transaction Processing Performance Council. *TPC-C*. 2013.
[4] H Hacigu¨mu¨ s, B Iyer, S Mehrotra. *Providing Database as a Service.* Proc. 18th IEEE Int'l Conf. Data Eng. 2002.
[5] L Ferretti, M Colajanni, M Marchetti. Distributed, concurrent, and independent access to encrypted cloud databases. *IEEE Transactionson Parallel and Distributed Systems.* 2013; 99: 2013.
[6] J Li, M Krohn, D Mazie`res, D Shasha. *Secure Untrusted Data Repository (SUNDR).* Proc. Sixth USENIX Conf. Opearting Systems Design and Implementation. 2004.
[7] Maha TEBAA, Saïd EL HAJJI, Abdellatif EL GHAZI. *Homomorphic Encryption Applied to the Cloud computing Securiy.* Proceedings of the World Congress on Engineering. London, U.K. 2000; 1.