AN EFFICIENT CONCURRENT ACCESS ON CLOUD DATABASE USING SECUREDBAAS

G.Elakkia
M.E., ComputerScience and Engineering
RVS technical campus, coimbatore
Anna University, Chennai.

Abstract: **In distributed database system, cloud services provide high availability and scalability, but they raise many concerns about data confidentiality. SecureDBaas guarantees data Confidentiality by allowing a cloud database server to execute concurrent SQL operations over encrypted data and the possibility of executing concurrent operations on encrypted data. It's 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. SecureDBaas that support the execution of concurrent and independent operations to the remote encrypted database from many geographically distributed clients. It is compatible with the most popular relational database servers, and it is applicable to different DBMS implementation. It provides guarantees to data confidentially by allowing a cloud database server to execute concurrent SQL operations over encrypted data.**

Keywords- **Database, SecureDBaaS, Cloud, Security**

1 INTRODUCTION

1.1 CLOUD COMPUTING SECURITY

Cloud security is an evolving sub-domain of computer security, network security, and, more broadly, information security. It refers to a broad set of policies, technologies, and controls deployed to protect data, applications, and the associated infrastructure of cloud computing. Cloud security architecture is effective only if the correct defensive implementations are in place. Efficient cloud security architecture should recognize the issues that will arise with security management. The security management addresses these issues with security controls. These controls are put in place to safeguard any weaknesses in the system and reduce the effect of an attack. While there are many types of controls behind cloud security architecture, they can usually be found in one of the following categories:

1.1.1 Deterrent controls

These controls are intended to reduce attacks on a cloud system. Much like a warning sign on a fence or a property, deterrent controls typically reduce the threat level by informing potential attackers that there will be adverse consequences for them if they proceed. [Some consider them a subset of preventive controls].

1.1.2 Preventive controls

Preventive controls strengthen the system against incidents, generally by reducing if not actually eliminating vulnerabilities. Strong authentication of cloud users, for instance, makes it less likely that unauthorized users can access cloud systems, and more likely that cloud users are positively identified.

1.1.3 Detective controls

Detective controls are intended to detect and react appropriately to any incidents that occur. In the event of an attack, a detective control will signal the preventative or corrective controls to address the issue. System and network security monitoring, including intrusion detection and prevention arrangements, are typically employed to detect attacks on cloud systems and the supporting communications infrastructure.

1.1.4 Corrective controls

Corrective controls reduce the consequences of an incident, normally by limiting the damage. They come into effect during or after an incident. Restoring system backups in order to rebuild a compromised system is an example of a corrective control.

1.2 CLOUD DATABASE

A database accessible to clients from the cloud and delivered to users on demand via the Internet from a cloud database provider's servers. Also referred to as Database-as-a-Service (DBaaS)[8], cloud databases can use cloud computing to achieve optimized scaling, high availability, multi-tenancy and effective resource allocation. While a cloud database can be a traditional database such as a MySQL or SQL Server database that has been adopted for cloud use, a native cloud database such as Xeround's MySQL Cloud database tends to better equipped to optimally use cloud resources and to guarantee scalability as well as availability and stability. Cloud databases can

offer significant advantages over their traditional counterparts, including increased accessibility, automatic failover and fast automated recovery from failures, automated on-the-go scaling, minimal investment and maintenance of in-house hardware, and potentially better performance.

Cloud DBMS (CDBMS) as a distributed database that delivers a query service across multiple distributed database nodes located in multiple geographically-distributed data centers, both corporate data centers and cloud data centers [6]. A query can originate from anywhere; from a PC within the corporation, which is connected by a fast line to the local data centre, from a PC in the home via a VPN line, from a laptop via a Wi-Fi connection, or from a smart phone via a 3G or 4G connection. For that reason we represent a query here as coming "through the Internet" implying that the response will possibly travel through the Internet too.

2 EXISTING SYSTEM

Existing System contain Proxy based cloud database, that requires any client operation should pass through one intermediate server but it is not suitable to cloud-based scenarios, in which multiple clients typically distributed among different locations, and need concurrent access to data stored in the same DBMS. If proxy fails then we cannot perform transaction between client and server and Policy inconsistencies during policy updates due to the consistency model.
Encrypting blocks of data instead of each data item [8]. Whenever a data item that belongs to a block is required, the trusted proxy needs to retrieve the whole block, to decrypt it, and filter out unnecessary data that belong to the same block. Then it requires heavy modification of the original SQL operation produced by each client. Thereby it causes significant overheads on both the DBMS server and the trusted proxy. Prevent one cloud provider to read its portion of data, but information can be reconstructed by cloud provider and allow Execution of operations over encrypted data. These approaches preserve data confidentiality, where DBMS is not trusted [3]. It requires modified DBMS software used by cloud provider. Proxy-less architectures that store metadata in the clients.

Some DBMS engine offers the possibility of encrypting data at the file system level. It means transparent data Encryption feature. This feature makes it possible to build a trusted DBMS over untrusted storage.

Focused on different usage contexts, including data manipulation, modification to the database structure and does not provide data confidentiality for database as a service. Trusted proxy that characterize and facilitates the implementation of a secureDBaaS, and is applicable to multitier web application. Its causes several drawbacks. Since the proxy is trusted, its functions cannot be outsourced to an untrusted cloud provider. Hence, the proxy is meant to be 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 single point of failure and a system bottleneck. Proxy is meant to implemented and managed by cloud tenant. There are several solutions ensuring confidentiality for the storage as a service by using sql aware encryption [13]:

3 PROPOSED SYSTEM

3.1 SECUREDBAAS

SecureDBaaS is designed to allow multiple and independent clients to connect directly to the untrusted cloud DBaaS without any intermediate server a cloud database service from an untrusted DBaaS provider [8]
SecureDBaaS architecture is tailored to cloud platforms and does not introduce any intermediary proxy or broker server between the client and the cloud provider.

Tenant then deploys one or more machines (Client 1 through N) and installs 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. 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 un trusted 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.

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. It is compatible with the most popular

relational database servers, and it is applicable to different DBMS implementations.

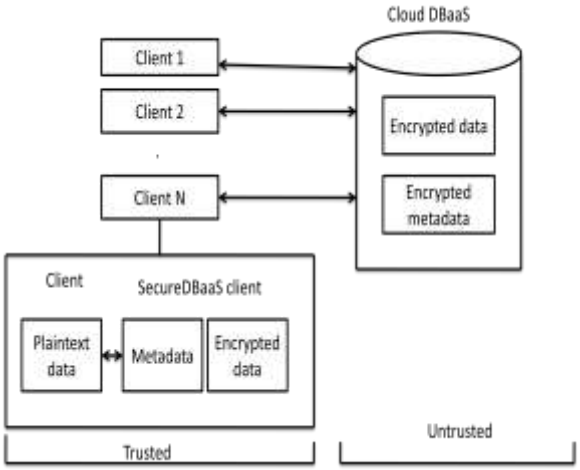

Figure 3.1 SecureDBaas Architecture

3.2 DATA AND METADATA MANAGEMENT

The tenants data are saved in a relational database 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. 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 Secure DBaaS 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 is 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.

 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.

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.

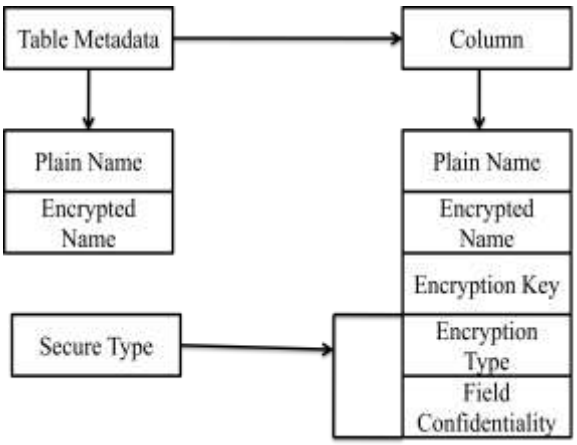

Figure 4.2 Structure of Table Metadata

3.3 SEQUENTIAL AND CONCURRENT SQL OPERATIONS

The SQL operations in SecureDBaaS by considering an initial simple scenario inwhich assume that the cloud database is accessed by one client. 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.

The support to concurrent execution of SQL statements issued by multiple independent (and possibly geographically distributed) clients is one of the most important benefits of SecureDBaaS with respect to state-of-the-art solutions. Our architecture must guarantee consistency among encrypted tenant data and encrypted metadata because corrupted or out-of-date metadata would prevent clients from decoding encrypted tenant data resulting in permanent data losses. A thorough analysis of the

possible issues and solutions related to concurrent SQL operations on encrypted tenant data and metadata is contained.

3.4 ALGORITHM

RSA(Rivest-Shamir-Adleman) widely accepted and implemented general-purpose approach to public-key encryption.

Plaintext is encrypted in block having a binary value less than some number. Both sender and receiver must know the value of n. The sender knows the values of e, and only the receiver knows the values of d. This is a public-key encryption algorithm with a public key of PU={e,n} and a private key of PR={d,n}.

Encryption and decryption are the following form, for some plaintext block M and cipher block C.

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n$

For this algorithm to be satisfactory for public-key encryption, the following requirement must be met.

- It is possible to find values of e,d,n such that $M^e \bmod n = M$ for all $M<n$.
- It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M<n$.
- It is infeasible to determine d given e and n.

Encryption by Bob with Alice's public key:

Plain text : $M<n$

Cipher text : $C= M^e \bmod n$

Decryption by Alice with Alice's Public key:

Cipher text : $M<n$

Plain text : $C= M^e \bmod n$

4 CONCLUSIONS

In the encrypted cloud database 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 heterogeneous and possibly geographically dispersed clients.

5 FUTURE ENHANCEMENTS

In this future work identifies consistency issues related to concurrent execution of queries over encrypted data and to propose viable solutions for different usage contexts, including data modification to the database structure, and data re-encryption. Data, policy, and credential inconsistency problems that can emerge as transactional database systems are deployed to the cloud. Data re-encryption context is to investigate that arise when clients re-encrypt data stored in the cloud database. This occurs when it is required to change encryption keys, or to use a different encryption algorithm to guarantee confidentiality. A re-encryption command that modifies the encryption key that is used to encrypt customer data stored in the table Tenc. The client first reads the current metadata (MR[T]) associated with the encrypted customer data to retrieve all the information related to their encryption policy, current encryption keys. Then, it updates the metadata (MW[T]) according to the new encryption policy, by changing the encryption keys. Hence, the client needs to read all the data (R[Tenc]), to decrypt them with the old encryption keys, to encrypt them with the new encryption keys and to write new data to the encrypted table (W[Tenc]). Decryption and encryption operations have to be performed locally by a trusted client because the client never exposes data to the untrusted cloud database.

Re-encryption and data read The database may return data that are not accessible by the client, if a data read command is executed concurrently to a re-encryption command. The case in which a data read command requires a set of data whose encryption key is being modified by a concurrent re-encryption command.

Re-encryption and data write Inconsistent data may be written if the data write command and a re-encryption command are executed concurrently. The case in which a data write command stores a set of data whose encryption key is being modified by a concurrent re-encryption command.

REFERENCES

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

[2] M. Armbrust et al., "A View of Cloud Computing," Comm. Of the ACM, vol. 53,no.4, pp. 50-58, 2010.

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

[4] A.J. Feldman, W.P. Zeller, M.J. Freedman, and E.W. Felten,"SPORC:Group Collaboration Using Untrusted Cloud Re-Sources," Proc. Ninth USENIX Conf. Operating Systems Design and Implementation, Oct. 2010.

[5] L. Ferretti, M.Colajanni, and M.Marchetti," Suppoting Security and Consistency for CloudDatabase," Proc. Fourth Int'I Symp.Cyberspace Safety and Security, Dec.2012.

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

[7] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over Encrypted Data in the Database-Service-Provider Model," Proc. ACM SIGMOD Int'I Conf. Management Data, June 2002.
[8] H. Hacigumus, B. Iyer, and S. Mehrotra,"Providing Database as a Service," Proc. 18th IEEE Int'l Conf. Data Eng., Feb.2002.
[9] J. Li,M .Krohn, D. Mazieres, and D. Shasha, "Secure Untrusted Data Repository (SUNDR)," Proc. Sixth USENIX Conf. Operating Systems Design and Implementation, Oct. 2004.
[10] 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.
[11] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin, and M. Walfish, "Depot: Cloud Storage with Minimial Trust," ACM Trans. Computer Systems, vol.29, no.4, article 12, 2011.
[12] 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.
[13] R.A. Popa, C.M.S. Redfield,N. Zeldovich, and H. Balakrishnan,"CryptDB: Protecting Confidentiality with Encrypted Query Processing," Proc.23rd ACM Symp. Operating System Principles, Oct. 2011.
[14] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar," An integrated Experimental Environment for Distributed System and Networks," Proc. Fifth USENIX Conf. Operating System Design and Implementation, Dec. 2002.

