International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Secure Cloud DBaaS by Client and Server side Encryption

Ashok Jayadar[1], A. R. Arunachalam[2]

[1]PG student, Department of Computer Science and Engineering, Bharath University, Chennai, Tamilnadu, India
[2]Assistant Professor, Department of Computer Science and Engineering, Bharath University, Chennai, Tamilnadu, India

Abstract: ***Storing confidential data in cloud must provide guarantee of security and availability of the data, even though many alternatives are existed for handling and storing of data, while information confidentiality solution for the dbase are still unfinished. The new architecture that integrate cloud storage service with data privacy and possess a feature of executing concurrent operations on encrypted data and along with the geographically distributed clients to connect directly to these cloud database which is encrypted and they also provided to execute their operations over the cloud database. This architecture eliminates the brokers (Intermediate proxies) it limits the scalability, elasticity, availability. High sensitive data are encrypted by RSA encryption and regular data are encrypted using AES technique so that overhead on the network can be minimized.***

Keywords: Secure DBaaS, RSA encryption, untrusted CDB, metadata, CSP.

1. Introduction

In cloud infrastructure system the critical data is placed cloud service provider which is third party services assuring the protection and confidentiality is the prior importance. This is the scenario where cloud and the third parties services has the availability of accessing the private information of the clients. So the original plain data is accessible by the trusted parties and remaining entrusted context data must be encrypted. There are several solutions ensuring the secrecy for the storage as service but still confidentiality of the dbaas is still in processing. The main advantage of the securedbaas has the solution of allowing cloud tenants to access the full advantage of dbaas qualities like availability, reliability, scalability, without providing or exposing the unencrypted data to the cloud providers. The architecture design is motivated by the multiple and independent clients to perform the operations on the encrypted data by the SQL statements, which can modify the database structure. Database as a service is an architecture and operational approach enabling IT providers to deliver database functionality as a service to one or more consumers.

The proposed architecture has the property of executing the independent and parallel operations to the remote encrypted database from any geographically located clients, as unencrypted database as service setup. In this system it is not including any intermediate proxy between the client and the cloud provider. Even after eliminating intermediate proxy it can achieve the same availability, elasticity and reliability of the DBaaS in cloud. The secure database as service is immediately applicable to any DBMS because it doesn't require any modifications to the cloud database. A large set of experiments supported real cloud platforms demonstrate that Secure DBaaS is straight away applicable to any package as a result of it needs no modification to the cloud information services. Overall this proposed architecture was providing security by using the cryptographic schema to the confidently information of clients and the con's are response time of the read/write operations will decrease as because the entire database is encrypted.

2. Trusted Proxy Based System

Some DBMS engines offer the possibility of encrypting data at the filesystem level through the so called Transparent Data Encryption. 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 it assumes that the cloud provider is untrusted. The confidence on a trusted proxy 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 secure DBaaS service are then bounded by availability, scalability, and elasticity of the trusted proxy, that becomes a single point of failure and a system bottleneck.

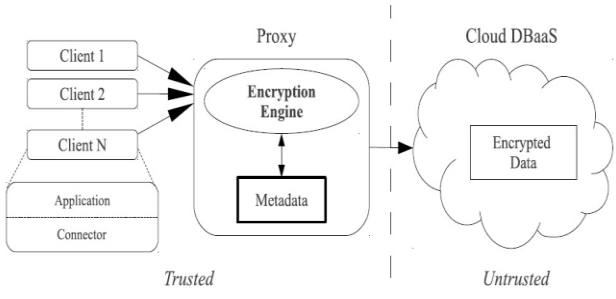

Figure 1: A proxy trusted Architecture

Licensed Under Creative Commons Attribution CC BY

3. Issues In Trusted System

A proxy-based architecture requiring that any client operation should pass through one intermediate server is not suitable to cloud based scenarios, in which multiple clients, typically distributed among different locations, need concurrent access to data stored in the same DBMS. It uses DES encryption and decryption which is not much powerful security in cloud database systems.

4. System Architecture

The System needs to extend to other platforms and to include new encryption algorithm with untrusted cloud database and trusted proxy needs to be removed. The virtual machine image on cloud uses cloud database independently. This can be achieved by using client application and cloud databases with RSA encryption engine for high security and AES encryption for regular data to fast access. The encrypted data is stored into the untrusted cloud database with encrypted metadata. Clouds do not need any trusted proxy for authentication and cloud database is authenticated as untrusted.

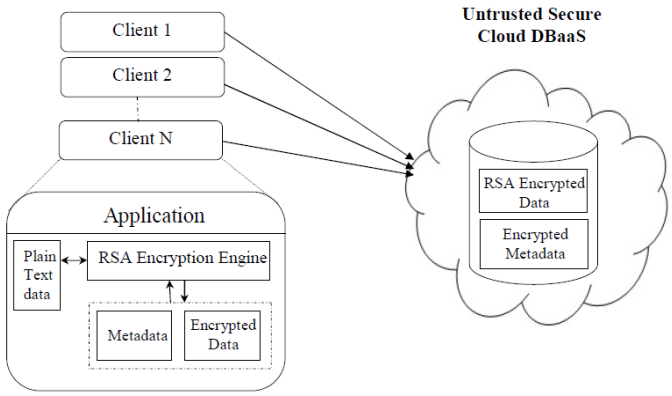

Figure 2: Architecture of untrusted secure RSA encrypted cloud DBaaS

5. Module Description

5.1 N - Client – System module uses 1 to N client database user which is used to encrypt data as RSA encryption. The JDBC used to connect with the cloud database. Client database is verified and authenticated before connection to the cloud database.

5.2 Encrypted metadata - The query writer is the software component that translates plaintext commands processed by the operation parser into SQL commands that will be executed by the untrusted cloud database over encrypted data. It leverages the encryption engine to execute all the encryption operations. Moreover, it interacts with the metadata manager to check whether all the operators contained into the plaintext commands are supported by the encryption policies applied to the relevant tenant data. Translated SQL commands are forwarded to the standard database connector.

5.3 Access to Client - SecureDBaaS proposes a different approach where all data and metadata are stored in the cloud database. SecureDBaaS clients can retrieve the necessary metadata from the untrusted database through SQL statements, so that multiple instances of the SecureDBaaS client can access to the untrusted cloud database independently with the guarantee of the same availability and scalability properties of run of the mill cloud DbaaS.

5.4 Verification and Authentication – Client user can login secure cloud database only after verification and Authentication for SQL execution like Create, Read, Update and delete data from cloud database. RSA involves a public key and private key. The public key can be known to everyone, it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The public key is made of the modulus n and the public (encryption) exponent e. The private key is made of the modulus n and the private (or decryption) exponent d which must be kept secret.

6. Flowchart

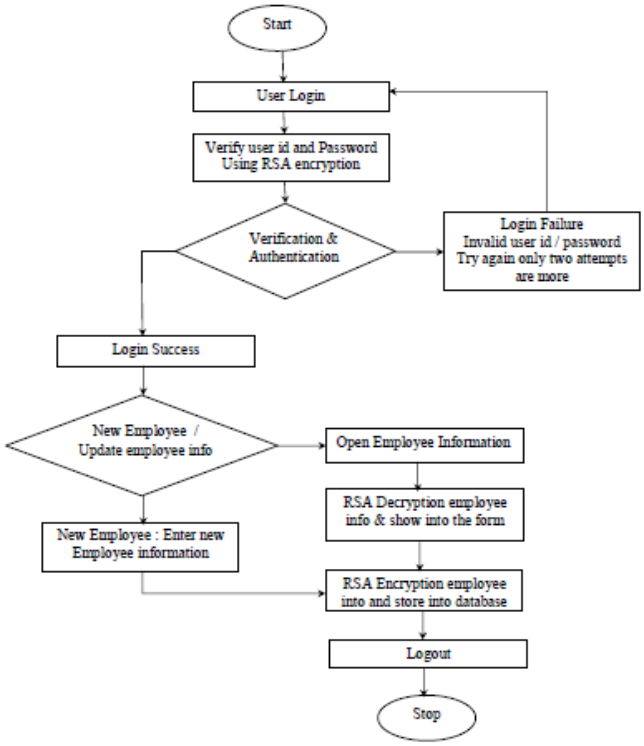

Figure 3: RSA encryption/decryption to access data

Licensed Under Creative Commons Attribution CC BY

7. Data Flow Diagram

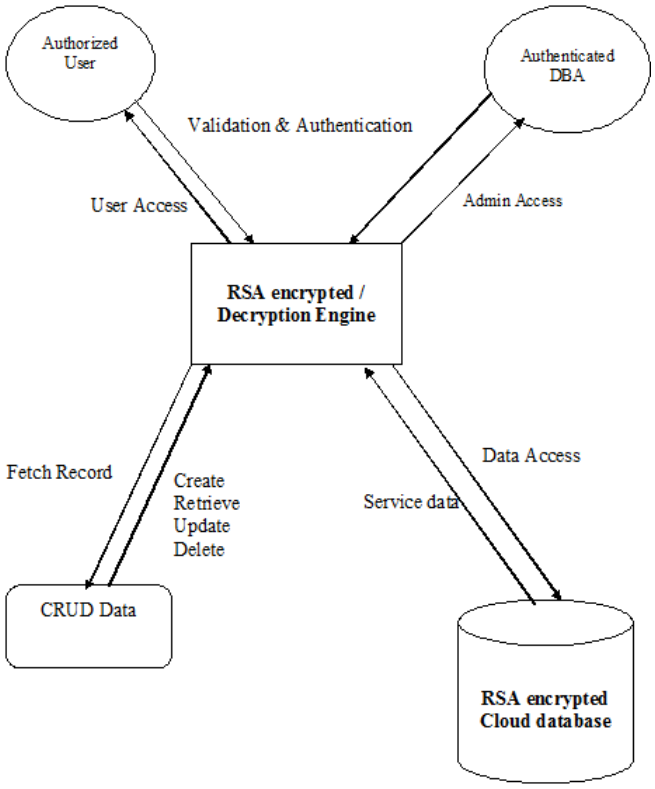

Figure 4: Data flow diagram for accessing data from cloud database

8. Usecase Diagram

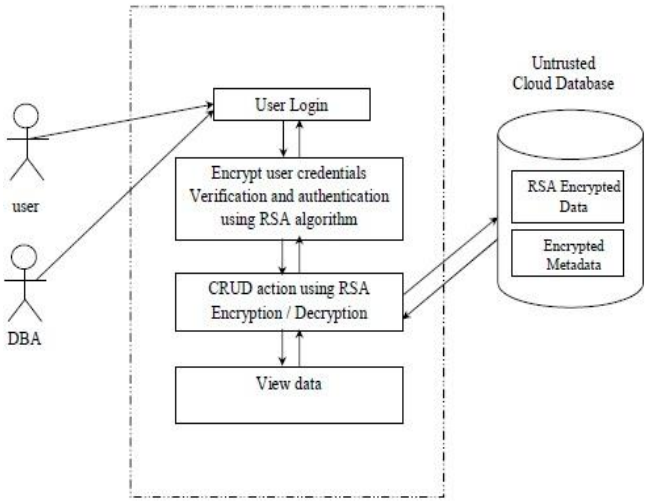

Figure 5: Usecase diagram to access data

9. Algorithm / Method Analysis

RSA involves a public key and private key. The public key can be known to everyone, it is used to encrypt messages. Messages encrypted using the public key can only be decrypted with the private key. The keys for the RSA algorithm are generated the following way:

1. Choose two different large random prime numbers *p* and *q*.
2. Calculate $n = pq$, *n* is the modulus for the public key and the private keys
3. Calculate the quotient: $\Phi(n) = (p - 1)(q - 1)$.
4. Choose an integer *e* such that $1 < e < \Phi(n)$, and ***e*** is co-prime to $\Phi(n)$ i.e.: *e* and $\Phi(n)$ share no factors other than 1; $gcd(e, \Phi(n)) = 1$. *e* is released as the public key exponent.
5. Compute *d* to satisfy the congruence relation $de \equiv 1 \pmod{\Phi(n)}$. i.e.: $de = 1 + k\Phi(n)$ for some integer *k*. *d* is kept as the private key exponent.

Descriptions about above algorithm:

Step 1: Numbers can be probabilistically tested for primarily.
Step 2: changed in PKCS#1 v2.0 to $\lambda(n) = lcm(p - 1),(q - 1)$ instead of $\Phi(n) = (p - 1)(q - 1)$.
Step 4: A popular choice for the public exponents is $e = 2^{16} + 1 = 65537$. Some applications choose smaller values such as $e = 3$, 5, or 35 instead. This is done to make encryption and signature verification faster on small devices like smart cards but small public exponents may lead to greater security risks.
Steps 4 and 5 can be performed with the extended Euclidean algorithm.

The public key is made of the modulus *n* and the public (or encryption) exponent *e*. The private key is made of the modulus *n* and the private (or decryption) exponent *d* which must be kept secret.

For efficiency a different form of the private key can be stored: *p* and *q*: the primes from the key generation, $d \bmod (p - 1)$ and $d \bmod (q - 1)$: often called *dmp1* and *dmq1*. $q^{-1} \bmod (p)$: often called *iqmp*. All parts of the private key must be kept secret in this form. *p* and *q* are sensitive since they are the factors of *n*, and allow computation of *d* given *e*. If *p* and *q* are not stored in this form of the private key then they are securely deleted along with other intermediate values from key generation.

Although this form allows faster decryption and signing by using the Chinese Remainder Theorem (CRT) it is considerably less secure since it enables side channel attacks. This is a particular problem if implemented on smart cards, which benefit most from the improved efficiency. (Start with $y = x^e \pmod{n}$ and let the card decrypt that. So it computes $y^d \pmod{p}$ or $y^d \pmod{q}$ whose results give some value *z*. Now, induce an error in one of the computations. Then $gdc(z - x, n)$ will reveal *p* or *q*.

10. Conclusions

The paper proposes a novel solution that guarantees confidentiality of data saved into cloud databases that are untrusted by definition. All data outsourced to the cloud provider are encrypted through RSA and AES cryptographic algorithms that allow the execution of standard SQL queries on encrypted data. This is one of the solution that allows direct, independent and concurrent

Licensed Under Creative Commons Attribution CC BY

access to the cloud database and that supports even changes to the database structure. It does not rely on a trusted proxy that represents a single point of failure and a system bottleneck, and that limits the availability and scalability of cloud database services. Concurrent read and write operations that do not modify the structure of the encrypted database are supported.

There are various encryption decryption techniques available and are having their limitations. The architectural design in this paper uses RSA algorithm which is highly secure for data, but RSA encryption may increase overheads, therefore to decrease the overhead in the network. Very important data are encrypted using RSA and remaining data are encrypted using AES. Specifically, simultaneous read and compose operations that don't adjust the structure of the encoded database cause unimportant overhead. Dynamic situations described by simultaneous adjustments of the database structure are upheld, however at the cost of high computational expenses. These execution effects open the space to future changes are exploring.

References

[1] Distributed, Concurrent, and Independent Access to Encrypted Cloud Databases, Luca Ferretti, Michele Colajanni, and Mirco Marchetti. IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 2, february 2014.
[2] Access control enforcement on query-aware encrypted cloud databases, Luca Ferretti, Michele Colajanni, and Mirco Marchetti . 2013 IEEE International Conference on Cloud Computing Technology and Science.
[3] Cloud Databases: A Paradigm Shift in Databases, Indu Arora and Dr. Anu Gupta. IJCSI Vol. 9, Issue 4, No 3, July 2012.
[4] Efficient Method to Secure Web applications and Databases against SQL Injection Attacks, Zeinab Raveshi, Sonali R. Idate IJARCSSE Volume 3, Issue 5, May 2013 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering.
[5] Secured Data Storage in Google Cloud S.SABARI VASAN, I.GOLDA SELIA, International Journal of Computer Trends and Technology (IJCTT) - volume4 Issue4 –April 2013.
[6] Handling Confidential Data on the Untrusted Cloud: An Agent-based Approach Ernesto Damiani, Francesco Pagano CLOUD COMPUTING 2010:International Conference on Cloud Computing, GRIDs, and Virtualization.
[7] Supporting Security and Consistency for Cloud Database L. Ferretti, M. Colajanni, and M. Marchetti CSS 2012, LNCS 7672, pp. 179–193, 2012. Springer-Verlag Berlin Heidelberg 2012.
[8] Privacy-preserving Mining of Association Rules from Outsourced Transaction Databases Fosca Giannotti, Laks V.S. Lakshmanan, Anna Monreale, Dino Pedreschi, and Hui (Wendy) Wang IEEE SYSTEMS JOURNAL VOL:7 NO:3 YEAR 2013.
[9] Transaction Management in Homogenous Distributed Real-Time Replicated Database Systems Ashish Srivastava, Udai Shankar, Sanjay Kumar Tiwari . IJARCSSE Volume 2, Issue 6, June 2012 ISSN: 2277 128X.
[10] Cloud Computing Service models, Types, Database and issues Rahul Bhoyar, Prof. Nitin Chopde IJARCSSE Volume 3, Issue 3, March 2013 ISSN: 2277 128X.

Author Profile

Ashok Jayadar is currently pursuing his Master's degree in Computer Science and Engineering at Bharath University, Chennai, India. He completed his M.CA degree (2008-2009) from Annamalai University, Chennai, India. His areas of interest are Secure Cloud Database management and Services, Web Services, Mobile computing and Cloud Computing. Email ashok.jayadar@gmail.com.

A R Arunachalam received his B.E degree in Computer Science and Engineering from Madras University in 2002 and received his M.Tech degree in Computer Science and Engineering from Bharath University in 2007.He is currently pursuing his Ph.D in Computer Science and Engineering at Bharath University, Chennai. He has 10 years of teaching experience and has guided many B.Tech and M.Tech projects.

Licensed Under Creative Commons Attribution CC BY