Proceedings of the 2014 Federated Conference on Computer Science and Information Systems pp. 885–889
DOI: 10.15439/2014F43
ACSIS, Vol. 2

Solution for Secure Private Data Storage in a Cloud

Kirill Shatilov, Vladislav Boiko, Sergey Krendelev, Diana Anisutina, Artem Sumaneev

Department of Information Technology, Novosibirsk State University, Novosibirsk, Russia

shatilov@ccfit.nsu.ru, boikovladislav@gmail.com, s.f.krendelev@gmail.com, diana.anisutina@gmail.com, sumaneevartem@gmail.com

***Abstract*—Cloud computing and, more particularly, cloud databases, is a great technology for remote centralized data managing. However, there are some drawbacks including privacy issues, insider threats and potential database thefts. Full encryption of remote database does solve the problem, but disables many operations that can be held on DBMS side; therefore problem requires much more complex solution and specific encryptions. In this paper, we propose a solution for secure private data storage that protects confidentiality of user's data, stored in cloud. Solution uses order preserving and homomorphic proprietary developed encryptions. Proposed approach includes analysis of user's SQL queries, encryption of vulnerable data and decryption of data selection, returned from DBMS. We have validated our approach through the implementation of SQL queries and DBMS replies processor, which will be discussed in this paper. Secure cloud database architecture and used encryptions also will be covered.**

I. Introduction

RAPID growth and development of cloud technologies has leaded them to popularity and widespread usage. Although customers are excited by cloud features and benefits, they are very concerned about confidentiality of data stored and processed in a cloud. Insider threats combined with a general lack of transparency into provider process and procedure has dropped confidence in security of cloud data storage [1].

Data confidentiality is highly important for Cloud Databases (Database as a Service, DbaaS), and there are threats of disclosure of vulnerable user's data to unauthorized parties. First of all, curious and malicious database administrators may capture or leak data [3]. Also a theft of database may possibly occur, leaving data in hands of malefactor [4].

Listed problems are still actual [2], and as a result multiple solutions to problem of trusting clouds have been developed. Encryption of all data in remote database was offered as a method of providing provable confidentiality [5, 6]. But such an approach demands all operations will be held on a client side after decryption of database content. Other solutions, such as MIT CryptDB [7], lack fully homomorphic encryptions and use third-party encryptions with relatively low cryptostrength and known vulnerabilities [8].

To address listed cloud security issues we designed secure cloud database architecture, several encryption algorithms and a SQL data encoding component. Proposed solution addresses mentioned challenges using following key ideas:

- The first is to parse SQL queries and encrypt selected user data on client side. None of encryption keys are passed to server or to any proxy; all confidential data passed to DBMS are secure.
- The second idea is usage of wide variety of order preserving and fully homomorphic encryptions. All encryptions that are used in proposed solution are proprietary developed encryptions with relatively high speed and provable cryptostrength.
- The third technique is a combination of various encryptions in single table. Encrypting different columns with different encryptions within one table greatly reduce chances of successful cryptanalysis.

In this paper we present designed architecture, description of main components and overview of used encryptions.

The next section of paper features a brief explanation of basic principles giving the main idea of a handling of SQL queries used in proposed solution. After it, in Section 3, we describe the general architecture of proposed secure cloud database and client's component. Section 4 gives some insight into encryptions, which are used in secure SQL queries processing. Finally, the last section of this paper summarizes our achievements, also exposing some possible future development and additional security features improving protection of cloud database.

II. Basic Principles

In this Section basic principles are discussed illustrating main idea of secure cloud database. Core component of proposed secure cloud database is SQL queries processor. All user's SQL queries are analyzed and transformed by program components on client's side before sending to DBMS (see Fig. 1). Similarly, all replies from database are processed.

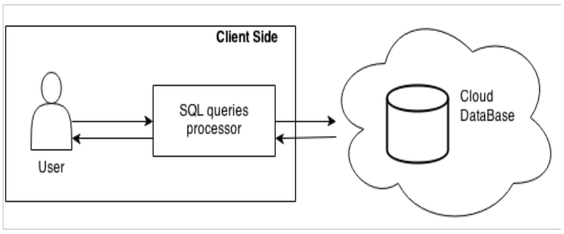

Fig. 1 Overview

978-83-60810-58-3/$25.00 © 2014, IEEE

Also, encryption of data, extracted from user's queries, is a responsibility of SQL query processing component. Proposed solution doesn't require any modifications on DBMS side.

User's SQL query handling is shown on Fig. 2. Information about encrypted columns such as encryption keys and encrypted column name(s) is needed in order to parse, decrypt and reconstruct user queries.

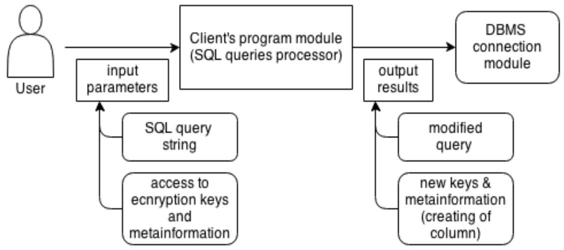
User
Client's program module (SQL queries processor)
DBMS connection module
input parameters
output results
SQL query string
access to ecnryption keys and metainformation
modified query
new keys & metainformation (creating of column)

Fig. 2 Query processing

When user retrieves data from cloud database, selection of column comes as a reply from DBMS. Information about encrypted columns is needed to decrypt and present selected data to user in suitable form (see Fig. 3)

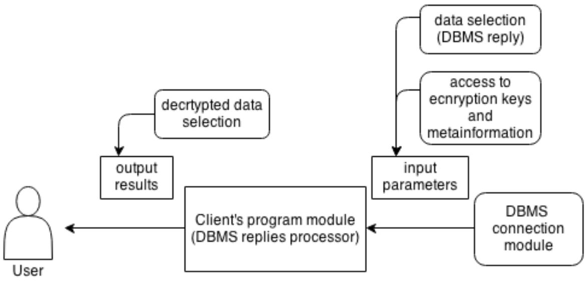
data selection (DBMS reply)
access to ecnryption keys and metainformation
decrtypted data selection
output results
input parameters
Client's program module (DBMS replies processor)
DBMS connection module
User

Fig. 3 Data selection processing

Basic idea of SQL queries processing is explained on different types of SQL expressions below.

"CREATE" statement is the only statement in terms of proposed secure cloud database, which may contain additional keywords, which are not included in SQL language. These keywords are markers of different encryptions applied to table columns. To indicate that column in currently created table should be encrypted, user should add a constraint corresponding to encryption's marker (identification string).

If SQL queries processing component encounters encryption marker, following steps will be performed. First, there encryption's keys are generated or chosen. Next, based on encryption information, number (can be more than one, in case when the result of encryption is a vector of multiple values), names, types and constraints of output columns are determined. Correct SQL string is created according to determined information.

After that modified statement is sent to DBMS. Output names of encrypted columns are anonymized, while anonymization of table name is optional (anonymization means changing real names of column to generated ones).

Processing of SQL statements such as "INSERT", "SELECT", "DELETE" uses common principles. All data from query are extracted and data from encrypted columns are encrypted. Also names of columns, which values are encrypted and used in processed statement, are modified according to their anonymized names. In some cases (for example, homomorphic encryption) changes in mathematical operation can be made. Responses for "SELECT" queries from database are decrypted if needed.

Correctness of performing "JOIN" operation inside DBMS depends on encryption properties. If encryption is deterministic, output column is single and both columns from each joining tables, have same key, no additional mechanisms are needed to perform "JOIN" operation.

It is very important to understand that some restrictions may apply to using full functionality of SQL language and different DBMS specific structures due to fact that proposed solution targets multi DBMS support, also various constraints can be caused by using order preserving or fully homomorphic encryption.

One of the restrictions is limited usage of "ALTER TABLE" construction is. As long as table altering doesn't affect encrypted columns, it can be performed, but adding or removing encryption from already existing table is unsupported. Another restriction is incompatibility of encryptions with several column constraints (e.g. "FOREIGN KEY").

This concludes basics principles and mechanisms of SQL queries processing in discussed approach. Main idea is to perform query analysis and modification, which include encryption of vulnerable user's data on client side, without affecting DBMS or adding any intermediate components.

III. ARCHITECTURE

Section 3 gives insight into secure cloud database architecture. As it was declared in previous sections, proposed solution does not use any DBMS components or any proxies in process of processing user's SQL queries and encrypting or decrypting data. All description of architecture applies to client's program module.

Client's program module consists of 4 basic components:

- Encryptions interfaces and encryption modules.
- Cryptographic metadata storage
- SQL queries processing component
- Database response's processor

Encryptions interfaces module provides two interfaces – "Key" and "Encoder". These interfaces define set of properties required for encryptions' correct work and interaction with other components. Due to "Encoder" and "Key" interfaces architecture and realization of entire solution does not depends on specific encryptions and is open to integration with other crypto algorithms.

Cryptographic metadata storage is responsible for storing information supporting SQL queries and DBMS replies processors. Among service information, the following values are kept in this storage:

- crypto keys for encryption of data in column
- map of real name of the column to anonymized names
- types of encryption used for column
- names of tables, where encrypted columns are located

Cryptographic metadata storage is an interface for retrieving and adding information about encrypted columns. Current realization means that all data are stored in file. File handling is a subject of future development. Expected solution is to store file encrypted on user's removable drives.

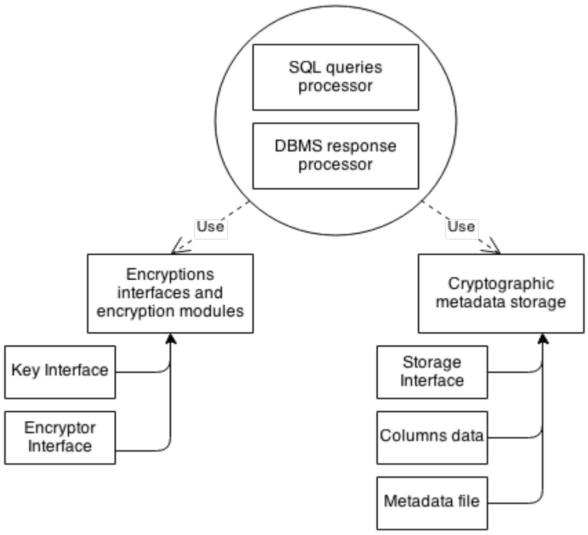

Fig. 4 Architectural overview

Core component of proposed solution is SQL queries processor; other components one way or another support its main function – analyze SQL queries and encrypt/decrypt data. SQL queries processing module consists of sub modules, each responsible for parsing exact SQL statement: "CREATE", "SELECT", "INSERT" and so on. These sub modules follow statements grammar to extract data, intended to be encrypted by user.

Last observed basic component is Database response's processor. Its main purpose is to detect encrypted columns in response, combine them (when multiple output columns correspond to single initial column), and to decrypt data to display them in suitable form to user. This module actively interacts with Cryptographic metadata storage, in order to correctly decrypt and modify response.

Architectural overview is summarized in Fig. 4. Two service components, Encryption and Crypto Storage modules, support SQL queries and database response processors in their main purpose – to manage all encryption and arithmetical transformations, while leaving secure cloud database's user with illusion of work with ordinary database.

IV. Encryptions

This section features description of encryptions that secure cloud database uses. We use three types of encryption: deterministic, order preserving and homomorphic encryptions.

Deterministic. Deterministic encryption provides strong security, it leaks only which encrypted values correspond to the same data value. In secure cloud database it can be used for storing password hashes, when no operations are conduct over data, but confidentiality is very important.

In proposed solution, we use proprietary developed deterministic block encryption [10].

Order Preserving. Order preserving (OP) encryption allows order relations between encrypted data items to be established, without revealing data itself. Such encryption can be used to protect salaries or other economical information inside secure database with possibility of performing order operations.

There are various OP encryptions, used in solution. For different types of data we can use different OP encryptions in single table; this provides extra resistance to crypto attacks. Only two encryption schemes will be discussed in this paper.

A. Arithmetic coding encryption

The first scheme, *based on arithmetic coding*, builds a representation of integer in an appropriate form. Assuming that integers are non-negative and do not require more than *n* bits, Then each number *c* is mapped to bit string $b=\left(a_1,a_2,\dots a_n\right)$ with first most significant bit. Let us define *f* as order-preserving mapping which maps string to some real number from interval $[0,1)$.

The simplest way to represent number *s*:

$$s=a_1\frac{1}{2}+a_2\frac{1}{2^2}+\dots+a_n\frac{1}{2^n}$$

One more way of representation:

$$s=\frac{c}{2^n}$$

We will seek for another representation for the number *s*.

We will use the arithmetic coding for representation f. Note, that *s* satisfies the equation:

$$2^n s=c$$

We will solve equation:

$$G(x)=2^n x-c=0 \qquad (1)$$

Equation (1) has only one solution on the interval $[0,1)$. In case of bisection method for seeking solution, the source number *s* will be found in *n* steps. The main idea of arithmetic coding is that intervals can be split at random. In this case, the approximate solution of (1) can be found in fewer steps which allow arithmetic coding to compress data.

Let us describe splitting process. Let define

$$k=\frac{p}{p+q},l=\frac{q}{p+q},k+l=1 \quad ,$$

where *p* and *q* are random natural numbers. Next, let us split interval $(0,1)$, in two pieces:

$$\left[0,\frac{p}{p+q}\right],\ \left[\frac{p}{p+q},1\right]$$

and calculate $G\left(\frac{p}{p+q}\right)$. If $G\left(\frac{p}{p+q}\right)>0$, we choose interval $\left[0,\frac{p}{p+q}\right]$ and return 0. Else if $G\left(\frac{p}{p+q}\right)<0$ then interval $\left[\frac{p}{p+q},1\right]$ is chosen and *l* is returned. Chosen interval will be marked with $\left[a_1,b_1\right]$.

New interval is split into two pieces at ratio $\frac{k}{l}$. After that following steps are performed: calculating value of *G(x)* in new point and choosing one of new intervals according to sign of *G(x)*; marking new interval with $\left[a_2,b_{12}\right]$. Proceeding by induction, we compute the interval $\left[a_n,b_n\right]$. Its length is $k^r l^{n-r}$, where *r* is the number of zeros in *b*. b_n is rational, so it can be expanded in powers of up to m degree, where m is the smallest number satisfying the equation:

$$\frac{1}{2^m}<k^r l^{n-r}$$

This equation can be rewritten in

$$-m<r\log_2 k+(n-r)\log_2 l$$

form, or

$$m>-r\log_2\frac{p}{p+q}-(n-r)\log_2\frac{q}{p+q}=$$
$$=r\log_2\frac{p+q}{q}+(n-r)\log_2\frac{p+q}{q}=$$
$$nlog_2 p+q-rlog_2 p-(n-r)\log_2 q$$

Therefore *m* can be calculated if *b*, *k* and *l* are known. Estimation of *m* does not have depended on *r* for the general case. There is very rough approximation:

$$m>n\log_2(p+q)$$

If for b_n there is an appropriate estimate which takes *m* bits, then bit string $f(b)=(b_1,b_2,\ldots,b_m)$. Designed mapping preserves the order by construction.

So, $f(b)$ is a value, that is sent to DBMS, the key is set of intervals (p_i,q_i) where value i is in interval $[1;n]$. Value *n* depends on size of input data.

B. Radix encryption

The second scheme's, *based on different number systems*, basic idea is conversion of numbers from notation with one radix to another.

For the first step is necessary to obtain the vector of coefficients from number in first-radix representation. Next step is replacing in the current representation first radix with second chosen radix. At the last step performed when a subsidiary vector of nonnegative numbers is added to the vector of coefficients from the current number representation. Note that values of the sum of these vectors must be less than second radix and second radix is greater than first.

Having final representation with new radix and modified coefficients the result can be calculated as second-radix – decimal conversion. The secret key is consist of first, second radixes and subsidiary vector of nonnegative numbers.

To illustrate this idea, let us consider one iteration of encryption. There can be made several iterations.

Original number is $s\in N$.

Secret key is $p,q\in N$, $\langle b_{0,} b_{1,}\ldots b_{n-1}\rangle$, $b_i<q,b_i\in N$.

Steps of encryption:

1. Obtaining S's p-radix representation:
$$s=\alpha_0+\alpha_1*p+\alpha_2*p^2+\ldots+\alpha_{n-1}*p^{n-1}$$
2. Replacing p-radix with q-radix:
$$s'=\alpha_0+\alpha_1*q+\alpha_2*q^2+\ldots+\alpha_{n-1}*q^{n-1}$$
3. Adding subsidiary vector $\langle b_{0,} b_{1,}\ldots b_{n-1}\rangle$ to the vector of coefficients
$\langle\alpha_0,\alpha_1,\alpha_2,\ldots,\alpha_{n-1}\rangle$.
$$s''=(\alpha_0+b_0)+(\alpha_1+b_1)*q+(\alpha_2+b_2)*q^2+\ldots$$
$$+(\alpha_{n-1}+b_{n-1})*q^{n-1}$$

where $\forall i:\alpha_i+b_i<q$.

The result of encryption is $w=s''$.

Process of decryption consists of following steps:

1. Obtaining w 's p-radix representation:
$$w=y_0+y_1*q+y_2*q^2+\ldots+y_{n-1}*q^{n-1}$$
2. Replacing q-radix with p-radix:
$$w'=y_0+y_1*p+y_2*p^2+\ldots+y_{n-1}*p^{n-1}$$
3. Subtracting vector $\langle b_{0,} b_{1,}\ldots b_{n-1}\rangle$ from the vector of coefficients $\langle y_{0,} y_{1,}\cdots y_{n-1}\rangle$:
$$w''=(y_0-b_0)+(y_1-b_1)*q+$$
$$+(y_2-b_2)*q^2+\ldots+(y_{n-1}-b_{n-1})*q^{n-1}$$

The result of decryption is w'', which is equal to s

The algorithm of encryption is correct and order preserving.

Modification of the considered scheme was used in the implementation. There are several iterations; also number of bits for values in the key can be specified in encryption module configuration.

This encryption has passed multiple tests and following results were measured:

Speed of encryption 125 Mbit / s

Speed of decryption 111 Mbit / s

(PC's configuration: Mobile Dual Core Intel Atom N570, 1666 MHz, 4 GB RAM, OS Windows 7).

Homomorphic. An encryption scheme is called fully homomorphic if it's able to evaluate an arbitrary function over ciphertexts. In this case decrypted value must match to a calculation result of the same function over plaintexts. The main feature of scheme [11] that is used in proposed secure cloud database is ability to define a strict upper bound of ciphertext size when performing calculations on it for both addition and multiplication.

V. Achievements and future Work

Proposed solution is not theoretical work only. We have implemented described crypto algorithms, measured encryption and decryption speed, optimized realization. Furthermore, prototype of client's program module has been successfully coded and tested. We used C++ and Boost's regular expressions to perform all described operations on SQL strings and databases' data selection. During tests on simple data scheme, with different "SELECT", "UPDATE", "DELETE" queries, slight (around 10-15%) overhead was detected, because of time elapsed for encryption/decryption.

Future development of proposed solution aims three main targets. First, is to develop and improve existing encryptions. Speed optimization is one of the most significant goals. Second target is to further develop of client's program module. This target includes functionality expansion, support of different DBMS, development of supporting modules (metafile encryption, authorization module).

The third aim is security improvement. Interest in analysis of encryption weaknesses and vulnerabilities [8, 9] is escalating, thus several measures can be taken to minimize risk of successful security breach. For example, to complicate frequency analysis, subsystem of phantom "SELECT" queries can be made in order to average number of queries to each column. Another idea for improving system's resistance to attacks is to add to columns garbage data that will be detected and ignored during decryption on client side. This method can change distribution of encrypted data massive inside DBMS, and as a result can make more difficult crypto attack on OP encrypted columns.

References

[1] Cloud Security Alliance. Top Threats to Cloud Computing V1.0 Cloud Security Alliance 2010.

[2] Cloud Security Alliance. The Notorious Nine. Cloud Computing Top Threats in 2013. Available: https://downloads.cloudsecurityalliance.org/initiatives/top_threats/The_Notorious_Nine_Cloud_Computing_Top_Threats_in_2013.pdf

[3] William R Claycomb, Alex Nicoll: Insider Threats to Cloud Computing: Directions for New Research Challenges CERT 2012.

[4] Privacy Rights Clearinghouse. Chronology of data breaches. Available: http://www.privacyrights.org/data-breach

[5] A. J. Feldman, W. P. Zeller, M. J. Freedman, and E. W. Felten. SPORC: Group collaboration using untrusted cloud resources. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[6] P. Mahajan, S. Setty, S. Lee, A. Clement, L. Alvisi, M. Dahlin and M. Walfish. Depot: Cloud storage with minimal trust. In Proceedings of the 9th Symposium on Operating Systems Design and Implementation, Vancouver, Canada, October 2010.

[7] R. A.Popa, C.M.S.Redeld, N.Zeldovich, and H.Balakrishnan: CryptDB: Protecting Condentiality with Encrypted Query Processing proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, 2011.

[8] L. Xiao, O. Bastani, I-Ling Yen: Security Analisys for Order Preserving Encryption Schemes, January, 10, 2012.

[9] R. Steinwandt, W. Geiselmann, and R. Endsuleit, "Attacking a polynomial-based cryptosystem: Polly Cracker," International Journal of Information Security, vol. 1, no. 3, pp. 143–148, 2002.

[10] Egorova V., Chechulina D., &Krendelev S. F. (2013) New View on Block Encryption (Unpublished) Available: https://db.tt/vnE9wfgj

[11] A A Zhirov, A., Zhirova, O., & Krendelev, S. F. (2013). Practical Fully Homomorphic Encryption over Polynomial Quotient Rings. In WorldCIS'13. London, UK.