ADAPTIVE ENCRYPTION TECHNIQUE FOR SECURITY AND DATA CONFIDENTIALITY SOLUTIONS IN CLOUD DATA BASE SERVICES

Dr. A. Bhuvaneswari*

M. Pradeepa**

Abstract: *The user's perception that the confidentiality of their data is endangered by internal and external attacks is limiting the diffusion of public cloud database services. In this context, the use of cryptography is complicated by high computational costs and restrictions on supported SQL operations over encrypted data. Database as a Service paradigm (DBaaS) poses several research challenges in terms of security and cost evaluation from a tenant's point of view. Cost models associated with different application execution models are proposed and these cost models can be composed to determine costs of different scenarios. We present techniques to estimate costs for service dependency and to monitor costs associated with typical scientific applications Database service model provide users power to create, store, modify and retrieve data from anywhere in the world as long as they have access to the internet. It introduces several challenges, an important issue being data privacy. Here it has been proposed that it takes advantage of adaptive encryption mechanisms to guarantee at runtime the best level of data confidentiality for any type of SQL operation. The adaptive encryption of public cloud database offers an interesting alternative to the trade-off between the required data confidentiality level and the flexibility of the cloud database structures at design time. The experiments are to be conducted out to show that these encryption schemes represent a feasible solution for achieving data confidentiality in public cloud databases even from a performance point of view.*

*Professor, CSE, Adhiparasakthi Engineering College

**PG Scholar, Adhiparasakthi Engineering College

I INTRODUCTION

The Database as a Service (DBaaS) is a novel paradigm through which cloud providers offer the possibility of storing data in remote databases. The main concerns that are preventing the diffusion of DBaaS are related to data security and confidentiality issues. Hence, the main alternative seems to be the use of cryptography, which is an already adopted solution for files stored in the cloud, but that represents an open issue for database operations over encrypted data. Fully homomorphic encryption theoretically supports any kind of computation over encrypted data, but it is computationally unfeasible, because it increases the computational cost of any operation by many orders of magnitude. Other schemes which allow the execution of computations over encrypted data limit the type of allowed operations (e.g., order comparisons in, sums in, search in). Although these methods were successfully deployed in some DBaaS contexts, they require the anticipatory choice of which encryption scheme can be used for each database column and for a specific set of SQL commands. In this paper, we propose cloud database architecture based on adaptive encryption techniques that encapsulate data through different layers of encryption. This adaptive encryption architecture is attractive because it does not require to define at design time which operations are allowed on each column, and because it can guarantee at runtime the maximum level of data confidentiality for different SQL operations. Unfortunately, this scheme is affected by high computational costs. However, through a prototype implementation of an encrypted cloud database, we show that adaptive encryption can be well applied to a cloud database paradigm, because most performance overheads are masked by network latencies. This study represents the first performance evaluation of adaptive encryption methods applied to cloud database services. Other experiments assumed a LAN scenario and no network latency.

II RELATED WORK

Improving the confidentiality of information stored in cloud databases represents an important contribution to the adoption of the cloud as the fifth utility because it addresses most user concerns. Our proposal is characterized by two main contributions to the state of the art: architecture and cost model. Although data encryption seems the most intuitive solution for confidentiality, its application to cloud database services is not trivial, because the cloud database must be able to execute SQL operations directly over encrypted data without accessing any decryption key. The tenant has two alternatives for any SQL operation : downloading the entire database, decrypting it, executing the query and, if the operation modifies the databases, encrypting and uploading the new data; decrypting temporarily the cloud database, executing the query, and re-encrypting it. The former solution is affected by huge communication and computation overheads, and costs that would make the cloud database services quite inconvenient; the latter solution does not guarantee data confidentiality because the cloud provider obtains decryption keys. The right alternative is to execute SQL operations directly on the cloud database, but avoiding that the provider obtains the decryption key. This proposal is based on data aggregation techniques associate plaintext metadata to sets of encrypted data to allow data retrieval. However, plaintext metadata may leak sensitive information and data aggregation introduces unnecessary network overheads. The use of fully homomorphic encryption would guarantee the execution of any operation over encrypted cloud data, but existing implementations are affected by huge computational costs to the extent that they would make impractical the execution time of SQL operations over a cloud database. Other encryption algorithms characterized by acceptable computational complexity support a subset of SQL operator.

III TECHNIQUES AND ALGORITHM

A. Adaptive Encryption Techniques

Consider SQL-aware encryption algorithms that guarantee data confidentiality and allow the cloud database server to carry out a large set of SQL operations over encrypted data. Each algorithm supports a specific subset of SQL operators. The following encryption schemes are used.

***Deterministic (Det*)**:** it deterministically encrypts data, so that the encryption of an input value always guarantees the same output value. It supports the equality operator.

***Order Preserving Encryption (OPE*)**: this encryption scheme preserves in the encrypted values, the numerical order of the original unencrypted data. It supports the following SQL operators: equal, unequal, less, less or equal, greater, greater or equal.

Sum: this encryption algorithm is homomorphic with respect to the sum operation: summing unencrypted data is equivalent to multiplying the correspondent encrypted values. It supports the sum operator between integer values.

Search: it supports equality check on full strings (i.e., the LIKE (operator) that do not include fragments of words.

Random (Rand): it is a semantic secure encryption (INDCPA) that does not reveal any information of the original plain value. It does not support any SQL operator.

Plain: a special kind of "encryption" that leaves values unencrypted. It supports all SQL operators, and it is included to store publicly available data, or some anonymous values that do not require any data confidentiality.

If each column data was encrypted through only one of these algorithms, the database administrator would have to decide at the design time which operations must be supported on each database column. This assumption is impractical in most cases. Hence, we need to define adaptive schemes that allow our architecture to support at runtime the SQL operations issued by the clients, while preserving a high level of confidentiality on the columns that are not involved in any operation.

B. Onion Layers

Organize the encryption schemes into structures called Onions. Each Onion is composed by different encryption algorithms, called (Encryption) Layers, one above the other. Outer Layers guarantee higher data confidentiality and lower number of allowed operations, and each Onion supports a specific set of operators. When additional SQL operations are to be executed on a column, the outer Layers are dynamically decrypted. In this paper, we consider and design the following Onions, which are also represented in Fig. 2.

Onion-Eq: it manages the equality operator.

Onion-Ord: it manages the following operators: less, less or equal, greater, greater or equal, equal, unequal.

Onion-Sum: it manages the sum operator.

Onion-Search: it manages the string equality operator.

Onion-Single-Layer: a special type of Onion that supports only a single Encryption Layer. It is recommended for columns in which operations to be supported are known at design time.

In our architecture, each plain database column is encrypted into one or more encrypted columns, each one corresponding to a different Onion, depending on the SQL operations that must be supported on that column. The most external Encryption Layer of an Onion is called Actual Layer, which by default corresponds to its strongest encryption algorithm.

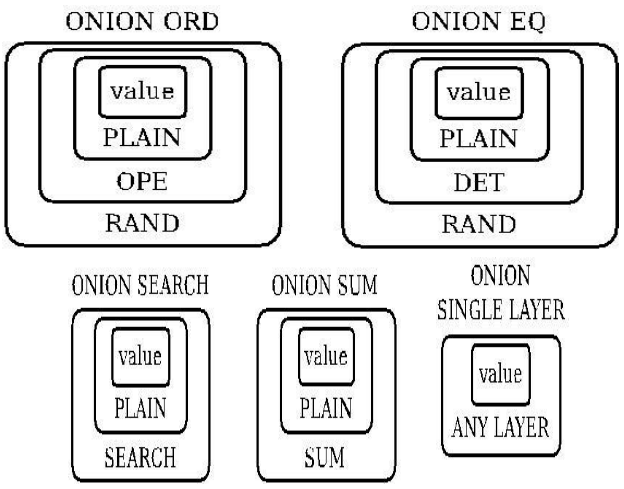


Figure 1: Onion layer and Structures

C. Cipher Block Chaining Algorithm

Cipher block chaining (CBC) is a mode of operation for a block cipher (one in which a sequence of bits are encrypted as a single unit or block with a cipher key applied to the entire block). Cipher block chaining uses what is known as an initialization vector (IV) of a certain length. One of its key characteristics is that it uses a chaining mechanism that causes the decryption of a block of cipher text to depend on all the preceding cipher text blocks. As a result, the entire validity of all preceding blocks is contained in the immediately previous cipher text block. A single bit error in a cipher text block affects the decryption of all subsequent blocks.

```
INPUT: plaintext x, key K
OUTPUT: cipher text y =
ASSUMED: round function g, last round h, key
scheduling procedure giving
for  i = 1 to e K(x)
Nr −1
Ki
w0= x
wi= g(wi−1,Ki), y = g(wNr−1,KNr−1)
```

D. Advanced Encryption Standard

1. Key Expansions

Round keys are derived from the cipher key using Rijndael's key schedule. AES requires a separate 128-bit round key block for each round plus one more.

2. InitialRound

AddRoundKey

Each byte of the state is combined with a block of the round key using bitwise xor.

3. Rounds

SubBytes

A non-linear substitution step where each byte is replaced with another according to a lookup table.

ShiftRows

A transposition step where the last three rows of the state are shifted cyclically a certain number of steps.

MixColumns

A mixing operation which operates on the columns of the state, combining the four bytes in each column.

4. Final Round (no MixColumns)

1. SubBytes
2. ShiftRows
3. AddRoundKey.

E. Blowfish Encryption

Blowfish is a symmetric block encryption algorithm designed in consideration with,

Fast **:** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.

Compact**:** It can run in less than 5K of memory.

Simple**:** It uses addition, XOR, lookup table with 32-bit operands.

Secure**:** The key length is variable ,it can be in the range of 32~448 bits: default 128 bits key length.

It is suitable for applications where the key does not change often, like communication link or an automatic file encryptor.

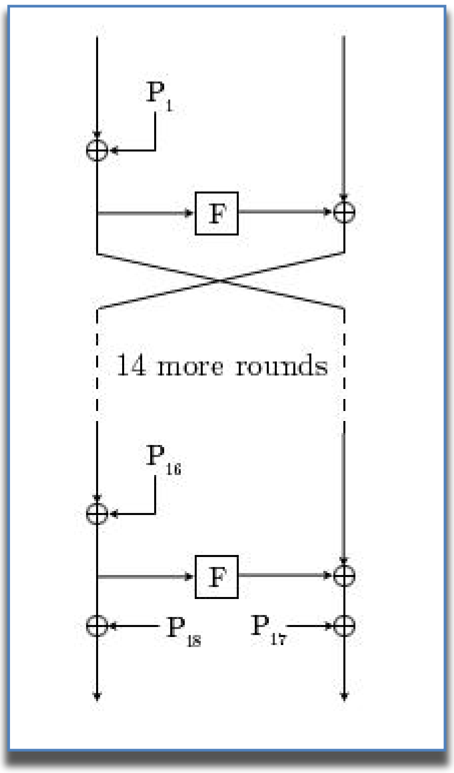


Figure 2: The Feistel structure of Blowfish

Description of Algorithm

Blowfish symmetric block cipher algorithm encrypts block data of 64-bits at a time.it will follows the feistel network and this algorithm is divided into two parts.

1. Key-expansion
2. Data Encryption

Key-expansion:

It will converts a key of at most 448 bits into several subkey arrays totaling 4168 bytes. Blowfish uses large number of subkeys.

These keys are generate earlier to any data encryption or decryption.

The p-array consists of 18, 32-bit subkeys:

P1,P2,………….,P18

Four 32-bit S-Boxes consists of 256 entries each:

S1,0, S1,1,………. S1,255

S2,0, S2,1,……….. S2,255

S3,0, S3,1,……….. S3,255

S4,0, S4,1,..............S4,255

Generating the Subkeys

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3):

P1 = 0x243f6a88, P2 = 0x85a308d3,

P3 = 0x13198a2e, P4 = 0x03707344, etc.

2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)
3. Encrypt the all-zero string with the Blowfish algorithm, using the subkeys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified subkeys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

   In total, 521 iterations are required to generate all required subkeys. Applications can store the subkeys rather than execute this derivation process multiple times.

Data Encryption

It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words.

The only additional operations are four indexed array data lookup tables for each round.

Algorithm: Blowfish Encryption

```
Divide x into two 32-bit halves: xL, xR
    For i = 1to 16:
        xL = XL XOR Pi
        xR = F(XL) XOR xR
```

```
            Swap XL and xR
    Swap XL and xR (Undo the last swap.)
            xR = xR XOR P17
     xL = xL XOR P18
      Recombine xL and xR
```

F. Paillier Algorithm

Choose two large prime numbers *p* and *q* randomly and independently of each other such that

$$\gcd(pq,(p-1)(q-1))=1$$

This property is assured if both primes are of equal length

Compute

$n=pq$ and

$\lambda=\operatorname{lcm}(p-1,q-1)$.

Select random integer g where

$$g\in\mathbb{Z}^*_{n^2}$$

Ensure n divides the order of g by checking the existence of the following modular multiplicative inverse:

$$\mu=(L(g^\lambda \bmod n^2))^{-1} \bmod n$$

where function L is defined as $L(u)=\frac{u-1}{n}$.

The notation $\frac{a}{b}$ does not denote the modular multiplication of a times the modular multiplicative inverse of b but rather the quotient of a divided by b, i.e., the largest integer value $v\geq 0$ to satisfy the relation $a\geq vb$.

The public (encryption) key is (n,g).

The private (decryption) key is (λ,μ).

If using p,q of equivalent length, a simpler variant of the above key generation steps would be to set

$$g = n + 1, \lambda = \varphi(n)$$ and

$$\mu = \varphi(n)^{-1} \bmod n$$, where

$$\varphi(n) = (p-1)(q-1)$$

Encryption

1. Let m be a message to be encrypted where $m \in \mathbb{Z}_n$
2. Select random r where $r \in \mathbb{Z}_n^*$
3. Compute ciphertext as: $c = g^m \cdot r^n \bmod n^2$

Decryption

1. Let c be the ciphertext to decrypt, where $c \in \mathbb{Z}_{n^2}^*$
2. Compute the plaintext message as:

   $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

IV ARCHITECTURE

The architecture is proposed to guarantee data confidentiality through adaptive encryption methods in cloud database environments.

Architecture Model

The distributed architecture represented in fig.2 where it is assumed that independent and distributed clients (Client 1 to N) access a public cloud database service. All information (i.e., data and metadata) is stored after encryption in the cloud database. The proposed architecture manages five types of information.

• ***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 are stored in the cloud database.

• ***Master key***: The encryption key of the encrypted metadata.

It is assumed to distribute all the above information to all the legitimate clients. A legitimate client can issue SQL operations (SELECT, INSERT, UPDATE, DELETE) to the encrypted cloud database by executing the following steps. It retrieves encrypted metadata, and obtains plain metadata by decrypting them through the master key.

The metadata are cached locally in a volatile representation that is used for improving performance. Then, the client can issue SQL operations over the encrypted data (i.e., the real informative content), because it is able to encrypt the queries, their parameters, and decrypt their results by using the local plain metadata. This architecture guarantees confidentiality of data in a security model in which the WAN network is untrusted (malicious), while client users are trusted- they do not reveal any information about plain data, plain metadata, and the master key. The cloud provider administrator is semi- honest (also called honest-but-curious), because he could try accessing information stored in the database, but he does not modify any internal data and SQL operations results.

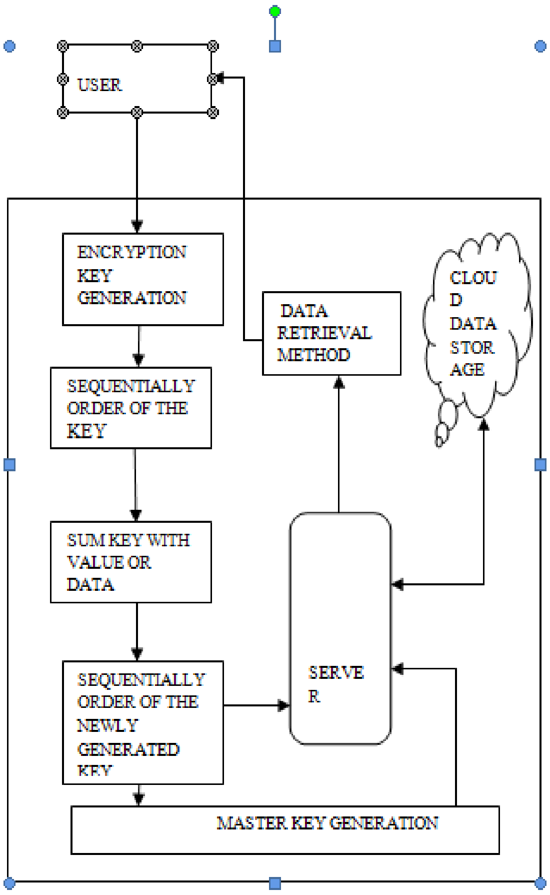

USER
ENCRYPTION KEY GENERATION
SEQUENTIALLY ORDER OF THE KEY
SUM KEY WITH VALUE OR DATA
SEQUENTIALLY ORDER OF THE NEWLY GENERATED KEY
DATA RETRIEVAL METHOD
CLOUD DATA STORAGE
SERVER
MASTER KEY GENERATION

Figure 3 : Cloud Database Architecture

V SYSTEM MODULES

A. Create Cloud Environment

Create a Cloud environment for cloud users using Cloud Data centers, Virtual Machines. Registration, login process is used to validate the clients in this module. Ubuntu operating system is used as cloud server and apache as web services and MYSQL as database services. Ubuntu is the most popular operating system on the leading public clouds and the fastest growing scale-out platform in the world. Ubuntu's goal is to secure "out-of-the box". By default user's programs run with low privileges and cannot corrupt the operating system or other user's files.

B. Homomorphic Encryption

Develop and Implement the cloud environment with homomorphic Encryption scheme for Virtual Machines. Dynamic resource allocation will help to allocate the resources with Encrypted data. Homomorphic encryption theoretically supports any kind of computation over encrypted data, but it is computationally infeasible, because it increases the computational cost of any operation by many orders of magnitude. Homomorphic encryption would guarantee the execution of any operation over encrypted cloud data, but existing implementations are affected by huge computational costs to the extent that they would make the execution time of SQL operations over a cloud database impractical.

C. Adaptive Encryption

The proposed system supports adaptive encryption methods for public cloud database service, where distributed and concurrent clients can issue direct SQL operations. By avoiding an architecture based on one or multiple intermediate servers between the clients and the cloud database, the proposed solution guarantees the same level of scalability and availability of the cloud service .The scheme of the proposed architecture where each client executes an encryption engine that manages encryption operations. This software module is accessed by external user applications through the encrypted database interface. The adaptive encryption scheme, which was initially proposed for applications not refering to the cloud, encrypts each plain column
into multiple encrypted columns, and each value is encapsulated into different layers of encryption, so that the outer layers guarantee higher confidentiality but support fewer computation capabilities with respect to the inner layers.

D. Cost Evaluation

The cost of a cloud database service can be estimated as a function of three main parameters:

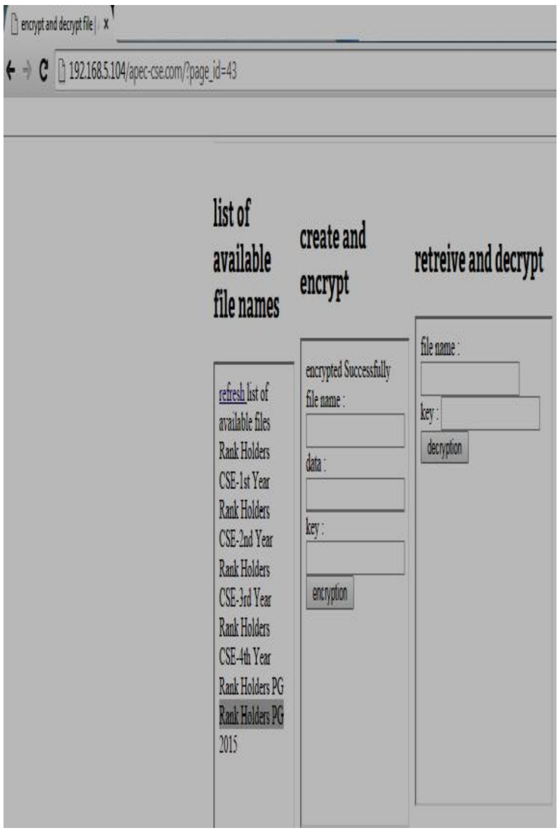
Cost = f(T ime, Pricing,Usage)

where:

- Time: identifies the time interval T for which the tenant requires the service.
- Pricing: It refers to the prices of the cloud provider for subscription and resource usage; they typically tend to diminish during T.
- Usage: denotes the total amount of resources used by the tenant; it typically increases during T .

VI EXPERIMENTAL RESULTS

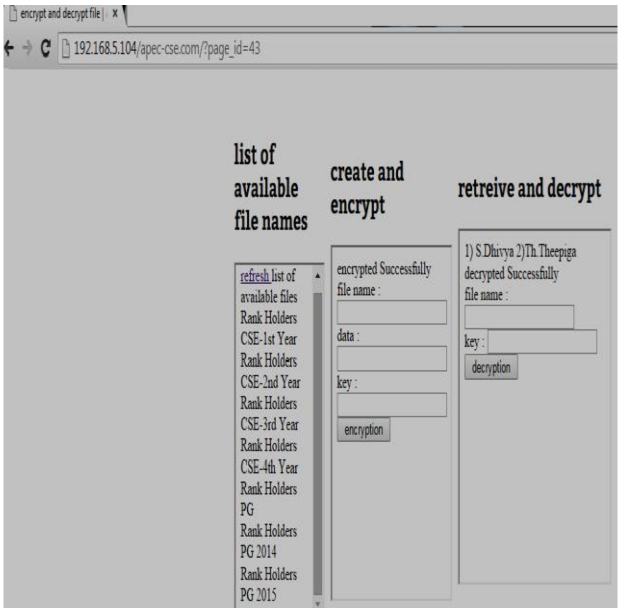
Encrypt the data using key with adaptive encryption techniques. First upload the data before encrypting the key. The encryption key is used to encrypt the metadata, after which it is stored in the cloud database. Finally the data is retrieved from the cloud after the decryption process is over. These figures will explain each process.



refresh list of available files
Rank Holders CSE-1st Year
Rank Holders CSE-2nd Year
Rank Holders CSE-3rd Year
Rank Holders CSE-4th Year
Rank Holders PG
Rank Holders PG 2015

encrypted Successfully

Figure 3: Enter File Name, Data And Key For Encryption



encrypt and decrypt file |

192.168.5.104/apec-cse.com/?page_id=43

list of available file names

refresh list of available files
Rank Holders CSE-1st Year
Rank Holders CSE-2nd Year
Rank Holders CSE-3rd Year
Rank Holders CSE-4th Year
Rank Holders PG
Rank Holders PG 2014
Rank Holders PG 2015

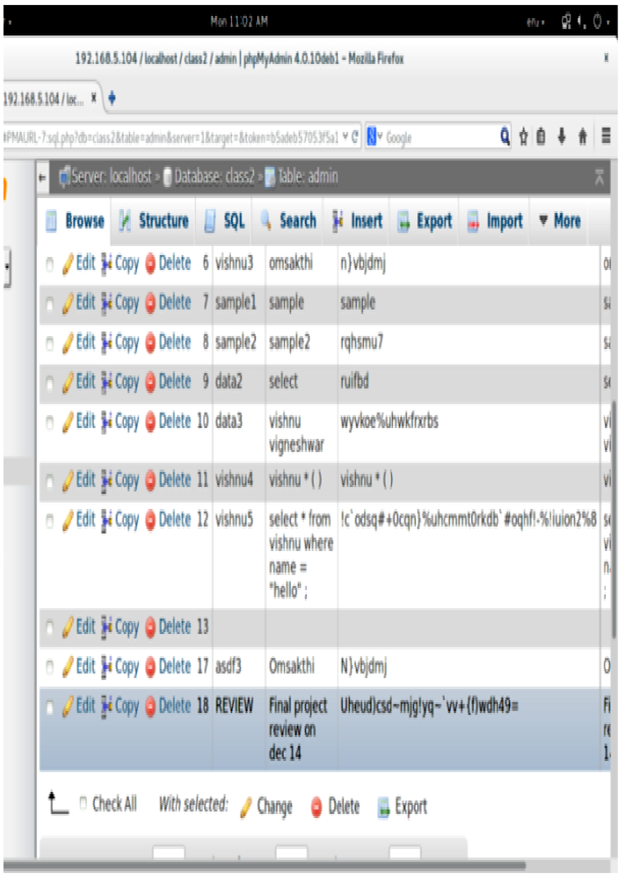
create and encrypt

encrypted Successfully
file name :
data :
key :
encryption

retreive and decrypt

1) S.Dhivya 2)Th.Theepiga
decrypted Successfully
file name :
key :
decryption

Figure 4: Original Data Is Retrive



Mon 11:02 AM

192.168.5.104 / localhost / class2 / admin | phpMyAdmin 4.0.10deb1 - Mozilla Firefox

#PMAURL-7:sql.php?db=class2&table=admin&server=1&target=&token=b5adeb57053f5a1

Server: localhost » Database: class2 » Table: admin

Browse | Structure | SQL | Search | Insert | Export | Import | More

Edit Copy Delete	6	vishnu3	omsakthi	n}vbjdmj	o
Edit Copy Delete	7	sample1	sample	sample	s
Edit Copy Delete	8	sample2	sample2	rqhsmu7	s
Edit Copy Delete	9	data2	select	ruifbd	s
Edit Copy Delete	10	data3	vishnu vigneshwar	wyvkoe%uhwkfrxrbs	vi vi
Edit Copy Delete	11	vishnu4	vishnu * ( )	vishnu * ( )	vi
Edit Copy Delete	12	vishnu5	select * from vishnu where name = "hello" ;	!c`odsq#+0cqn}%uhcmmt0rkdb`#oqhf!-%!uion2%8	s vi n ;
Edit Copy Delete	13				
Edit Copy Delete	17	asdf3	Omsakthi	N}vbjdmj	O
Edit Copy Delete	18	REVIEW	Final project review on dec 14	Uheud)csd~mjg!yq~`vv+(f)wdh49=	Fi re 1

Check All With selected: Change Delete Export

Figure 5: Store Filename, Encrypted Data In Cloud Database

Comparisons Between Existing and Proposed Algorithm

PARAMETERS	EXISTING	PROPOSED
ALGORITHM	Homomorphic Encryption	Adaptive Encryption
CPU UTILIZATION	Low	High
SECURITY	Security is not improved, because here used simple Encryption Algorithm.	Security is improved, because here used secure Encryption Algorithm.
COST	Cost is increased	Cost is decreased

VII DISCUSSION AND CONCLUSION

Data confidentiality architecture allows to leverage adaptive encryption mechanisms while avoiding the use of any intermediate (trusted) proxy server to manage encryption details. There are several benefits characterizing the proposed architecture. It guarantees confidentiality of information stored in the cloud database, while allowing the execution of SQL operations over encrypted data. It simplifies database configuration, because it does not require to define manually at design time which operations should be allowed on each column. It guarantees best level of data confidentiality for any SQL workload. It does not require any intermediate (trusted) proxy to manage encryption details. Adaptive encryption is also affected by two major drawbacks. The first problem is that each plain column must be encrypted into one or more encrypted columns (Onions), thus increasing the overall database size up to one order of magnitude. This cost may be considered acceptable, or it can be reduced by the database administrator through a suitable tuning of the confidentiality parameters. The second problem is the performance overhead characterizing adaptive encryption that has to encrypt all parameters and decrypt the results of every SQL operation through all the Encryption Layers of each involved Onion. These costs prevent the use of adaptive encryption methods on most real contexts. However, that this overhead becomes less significant when an encrypted database is used in the cloud, because in these scenarios realistic network latencies tend to mask the CPU time of expensive operations.

The Proposed architecture supports adaptive data confidentiality in cloud database environments without requiring any intermediate trusted proxy. Adaptive encryption mechanisms have two main benefits: they guarantee at runtime the maximum level of data confidentiality for any SQL workload, and they simplify database configuration at design time. If the workload is characterized by many similar operations, the present alternative is to accept this cost when data confidentiality is more important than performance. From the research point of view, it would be also inteesting to evaluate the proposed or alternative architectures under different threat model hypotheses.

REFERENCES

1. H.-L. Truong and S. Dustdar, "Composable cost estimation and monitoring for computational applications in cloud computing environments," Procedia Computer Science vol. 1, no. 1, pp. 2175 –2184, 2010, iCCS 2010.
2. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41,no. 1, pp. 23–50, 2011.
3. L. Ferretti, F. Pierazzi, M. Colajanni, and M. Marchetti, "Security and confidentiality solutions for public cloud database services,"in Proc. Seventh Int'l Conf. Emerging Security Information, Systems and Technologies, Aug. 2013.
4. T. Mather, S. Kumaraswamy, and S. Latif," Cloud security and privacy: an enterprise perspective on risks and compliance". O'Reilly Media, Incorporated, 2009.
5. O. Goldreich, Foundations of Cryptography: Volume 2, Basic Applications. Cambridge university press, 2004.
6. J. Daemen and V. Rijmen, The design of Rijndael: AES – the advanced encryption standard. Springer, 2002.
7. A. Fekete, D. Liarokapis, E. O'Neil, P. O'Neil, and D. Shasha, "Making snapshot isolation serializable," ACM Transactions on Database Systems, vol. 30, no. 2, June 2005, pp. 492–528.
8. R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "CryptDB: protecting confidentiality with encrypted query processing," in Proc. of the 23rd ACM Symposium on Operating Systems Principles, October 2011, pp. 85–100.
9. B. Moeller (May 20, 2004), Security of CBC Ciphersuites in SSL/TLS: Problems and Countermeasures.

10. Amazon RDS Pricing, “Amazon Relational DatabasePricing,”http://aws.amazon.com/rds/pricing, Mar. 2014.
11. 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 systems and networks,” in Proc. Fifth USENIX Conf. Operating Systems Design and Implementation, Dec. 2002.
12. TPC, “Tpc-c on-line transaction processing benchmark.” http://www.tpc.org/tpcc, Mar. 2014.
13. Amazon, “Amazon Web Services Blog,” http://aws.typepad. com/aws/price-reduction/, Mar. 2014.