# Practical System for Querying Encrypted Data on the Cloud

Parul Upadhyaya[1] Vimal Anjana[2] Vivek Jain[3] Marimuthu K.[4] Ganesh Gopal D.[5]
School of Computing Science and Engineering, VIT University
Vellore, TamilNadu - 632014, India
parul.upadhyaya2009@vit.ac.in, vimaljkasp@ymail.com, vivek.jain@ymail.com, k.marimuthu@vit.ac.in, ganeshgopal@vit.ac.in

**ABSTRACT**

**This paper proposes, compares and analyses query optimization techniques to securely query encrypted databases. Such querying capability is becoming increasingly relevant as individuals and enterprises move their data to the cloud. The schemes discussed delve into various design issues like speed, scalability and efficiency that are encountered in the context querying encrypted data that is on the cloud. We determine the exact differences in terms of the nature, scenario and behavior amongst the different approaches in between the existing probable solution and our solution that can be applied to the querying problem.**

KEYWORDS: cloud, security, speed, scalability, encryption, query

## I. INTRODUCTION

In today's era of data-driven computing, existence and usage of data management technology is inevitable. We deal with ever increasingly amount of data and information every day. The amount of data and information is increasing *at an exponential rate* [3]. Almost each and every sector involves data storage and processing over it. Recently, cloud computing systems are gaining popularity mainly due to the huge benefits [5] they provide such as ease of use, low upfront costs, low administration costs, high flexibility and very importantly the *virtualization of the present systems* [6]. The cloud is being used as a platform for storage, some popular examples being *Dropbox* [4] and *Amazon S3* [7], and efforts are being made to use cloud-based resources for querying, *manipulating and processing relational data* [2]. Many organizations host their databases on the cloud and use cloud-based computing resources. But one important concern for organizations in adopting the cloud is the security and privacy of their data, since they might want to keep the data private from the cloud not just to keep it private from the database administrators of the cloud but to also prevent information leakage in case of a security breach.

Formally, the existing technique of *encryption at rest* [4] explains that the working domain can be divided into two parts: a *trusted domain* and an *untrusted domain*. Traditional security models assume that given a computer system, any storage system (like hard disks) is an untrusted component while the other internal parts like main memory and CPU are taken as trusted. The data is stored only after encryption in untrusted domain and when any processing is required, the decryption is carried out in the trusted domain like main memory. Thus, although the data is stored in the untrusted domain but since it is in an encrypted form and the processing is carried out in the main memory (the trusted domain), the system becomes impenetrable for attackers from outside. But, in a cloud computing context, even the main memory and CPU are untrusted since when it comes to administrators or super privileged users, they can misuse their access rights of data. Thus, the only trusted domain is the data owner's local (non-cloud) machine.

In the current scenario, the querying technique that has been suggested requires shipping the entire encrypted data to the client site, decrypting it and then processing over it. That is, on receiving a request for data retrieval for the purpose of processing, the whole dataset is downloaded from the cloud instead of just the required subset of data. The processing is then carried on the dataset and then the data is again stored back on cloud. But this is very *expensive* and hence *does not scale to large datasets* [4][7]. To overcome these shortcomings we present a new scheme that attempts to evaluate as much of the query as possible on the cloud *without decrypting the whole data*. In practice, this scheme suggests shipping of partial data, querying on it and then shipping the required data corresponding to the searched parameter back from the whole dataset. Our approach reduces the expense of shipping the whole data while guaranteeing the security of data by encrypting the data in the untrusted domain.

In this paper, we propose a new approach: Instead of shipping the whole data to the trusted domain, only that subset should be shipped that is required to answer the query. According to the query, the relevant cells of the database is first shipped and searched for the required parameter and then based on the result the required data is retrieved from the database.

## II. PRELIMINARIES

We now formally define the key components of our system:

*Queries:* Our queries are the standard *SQL* queries that declaratively specify the data to retrieve. In this paper, we focus on the common case of Select-Project-Join queries.

*Database:* We only consider relational databases. This is a collection of organized as well as related relevant real world data.

*Encryption and Decryption:* The process of encoding a given string of text into another (called the cipher text) such that it is not easily understood by the eavesdroppers or hackers is known as encryption. Whereas decryption is the reverse of encryption in which the encoded text can be decoded back. Also, a private key is used to both encrypt and decrypt the text which is private for the owner and hence adds to the security. [1]

*Scalability:* It can be defined as the capability of a system, network or process to work with the growing amount of work as efficiently as it does for small amount of work or adapting to the growth in an efficient manner. [8]

Also we suggest the use of Advanced Encryption Standard (AES) to encrypt the dataset that is to be queried. It uses the plain text in block length of size 128 bits. Also the key can be of 128 bits or 192 bits or 256 bits. The number of rounds for each key size varies from 10, 12 and 14 for 128 bits, 192 bits and 256 bits, respectively. In each round, except for the last one, the following four steps are followed: substitute bytes, shift rows, mix columns and add round key. In the last step "mix column" step is not performed. The basic structure of AES looks like as shown in the figure 1 drawn below:

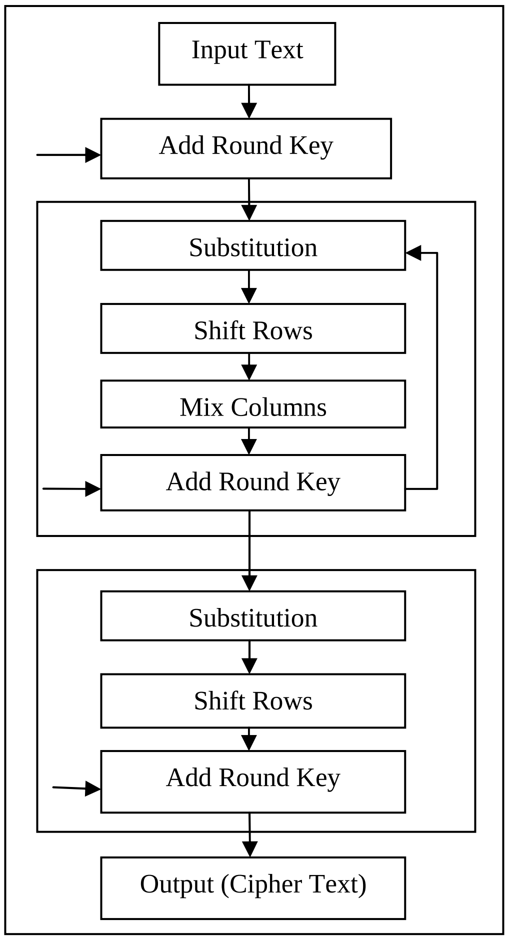

Figure 1 Basic Architecture of AES

The main reasons for using AES are as follows:

*Structure*: It does not follow a fiestal structure which is a symmetric structure used in making block ciphers. It uses the entire block length for operations.

*Security*: It uses the block length of 128 bits. And generally larger block length is preferred for more security.

*Robustness*: It goes through multiple rounds of permutation and combination that makes it very strong.

*Flexibility***:** It also provides different key lengths which make it much more flexible.

## III. METHODOLOGY

Our idea is to provide a more speedy method that would reduce overall cost of operation too. Our method has two main benefits: first, it reduces the latency of answers received; second, it reduces the monetary cost of using the cloud by reducing the amount of downloaded data. In our proposal, for accessing the data, the query is put forward in the usual way just like one does in the already existing proposition. The real difference lies in the part of the database that is worked on. Because of this our solution can transparently replace existing solutions with no change from user point of view. Unlike the already existing proposition where the whole dataset is shipped, decrypted and then queried, in this scheme, a particular column or field of the dataset (table) is accessed, shipped and matched against the keyword mentioned in the query. If a match is found, the dataset is again visited back and searched for the row that has that particular entry. The corresponding values are then retrieved. The whole process is diagrammatically represented in figure 2.

Say for an instance, we have 1000 entries in our local system database that we wish to store on the cloud using AES encryption we encrypt the data and store. Now we wish to access the data for carrying out some processing over it. With the proposed scheme, on the access command, particular column of the dataset is fetched and decrypted.

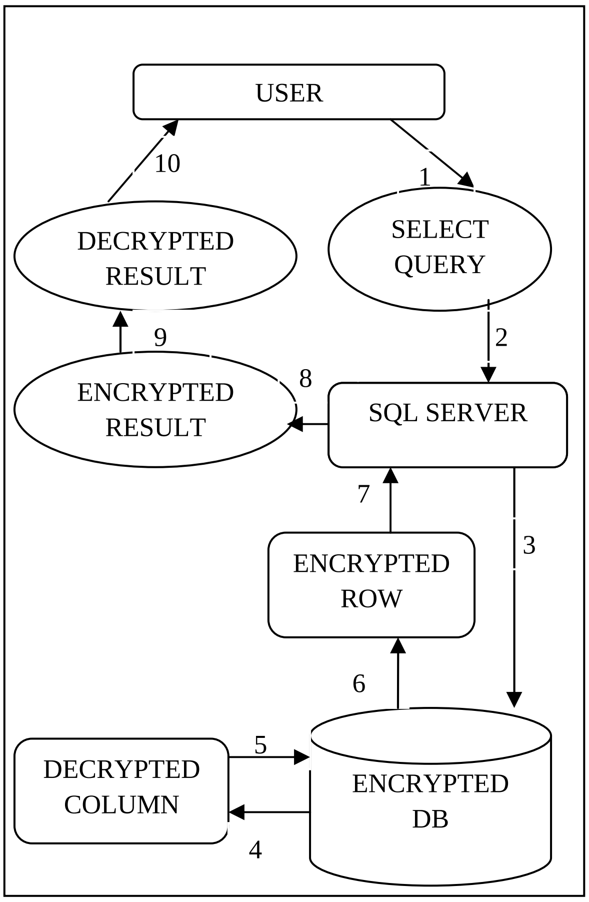

Figure 2 Diagrammatic representation of the proposed scheme

The following steps describe the steps as mentioned in the figure drawn above-

1. User enters a query
2. The query is transferred to SQL server
3. The Server queries the encrypted dataset present in the database.
4. The server decrypts the column according to the parameters present in the query
5. The query is again run over the decrypted column to find match for the corresponding required encrypted rows.
6. The required encrypted rows are fetched
7. The encrypted rows are transferred to the server.
8. SQL server returns the result in encrypted form.
9. The result is decrypted.
10. The decrypted result is transferred to the user at local system

## IV. ANALYSIS

In the most obvious solution to our problem of querying for the purpose of processing, the whole dataset needs to be downloaded for any kind of processing whether it is a big computational problem or small-scale computation. Even for making changes in just 1 entry, the whole dataset needs to be shipped. Whereas in the proposed scheme instead of shipping the whole dataset, initially just a part of it i.e. a particular chosen column is downloaded, searched against the keyword mentioned by user and accordingly, the corresponding row entries (records) are brought to the client system. In this way the overhead involved in shipping the data is spared and workload surely is cut down, that ultimately contributes to a more speedy system.

In terms of scalability if we take a very small dataset and employ the most obvious solution the time taken to transfer the whole dataset will be less but if we go on increasing the load by increasing the size of dataset, the time taken to ship data will get significantly increased. As a result of this the system will take more time for processing. In the proposed scheme we are just importing a column; even if size is increased load on the system is any time quite less in comparison to the importing of the whole dataset. To be more precise even if we go on increasing the number of fields (columns) there is no significant change in the taken to ship the data and hence the processing time does not get greatly increased. In short we always deal with 1 column irrespective of the number of columns present in the dataset. So we can conclude that the proposed scheme is far more scalable than the most obvious solution.

The efficiency of any system largely depends on the various above-mentioned factors like speed and scalability. So we have seen that the speed of the system with the newly proposed scheme is far more than that of the most obvious solution. Also the system using the proposed scheme is more scalable than that of the system using the most obvious solution. One more factor which leads to better efficiency is that there is less data transfer which leads to lesser data transfer time and more of computation time and also lesser system idle time and hence more system utilization. So these all factors lead to a more efficient system.

As clearly deducible cost of downloading a portion of dataset is far lesser than downloading the whole dataset, the system using the proposed scheme is cheaper than the one using the most obvious solution.

## V. CONCLUSION

By looking at the whole scenario and the analysis part, we can hereafter conclude that the proposed scheme is more scalable and faster with lesser downloading cost than the existing solution of querying encrypted data present in cloud.

## REFERENCES

[1] H. Lee Kwang, *Basic Encryption and Decryption*, Department of Electrical Engineering & Computer Science, KAIST

[2] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan, "*CryptDB: Protecting Confidentiality with encrypted Query Processing*", MIT CSAIL

[3] Fuxiong Wang, Ziqian Xiao, Jingyou Chen, " *Research on Security of Trusted Network and Its Prospects*", Department of Software Engineering, Hainan Software Profession Institute

[4] Arvind Arasul, Spyros Blanas, Ken Eguro, Raghav Kaushik, Donald Kossmann, Ravi Ramamurthy, Ramaratnam Venkatesan, " *Orthogonal Security With Cipherbase*", Microsoft Research, UW-Madison, ETH-Zurich

[5] Benefits of cloud computing available on :*http://www.salesforce.com/uk/socialsuccess/cloud-computing/why-move-to-cloud-10-benefits-cloud-computing.jsp*

[6] Importance of system virtualization available on: *http://www.2x.com/blog/2013/02/news/advantages-of-cloud-computing/*

[7] Amazon Web Services: http://*aws.amazon.com/s3*

[8] Details about scalability: *http://en.wikipedia.org/wiki/Scalability*