Elaine D mello

Be-IT

Roll no -11

Summary :A Comprehensive Framework for Secure Query Processing on Relational Data in the Cloud

Keywords: Data security in the cloud, Query processing on encrypted data,

Data confidentiality, Data availability.

Cloud computing has been gaining interests in the commercial arena due to its desirable features of scalability, elasticity, fault-tolerance, self-management and pay-per-use.Data confidentiality is one of the most important security concerns and challenges. Many existing proposals of query processing on encrypted data do not consider both confidentiality for data residing in storage and for data being accessed by queries [2, 3, 4, 5, 6, 7]. Second, different queries must be supported in the same framework, and practical query performance should not be lost in pursuit of the above data confidentiality requirements. Some of the previous works are only able to support one or two types of queries on encrypted data, and in general do not support data updates [3, 4]. The powerful cryptographic techniques such as homomorphic encryption [8] and Private Information Retrieval (PIR) [9] can satisfy the above mentioned data confidentiality requirements, but they are computationally expensive and can adversely impact both latency and ,throughput. The approaches improving the performance of PIR via the use of special hardwares [10, 11] may not be feasible for some small businesses who do not have the resources to make such investments. Next to data confidentiality are the concerns of data availability and integrity. Information Dispersal Algorithm (IDA) [12] and similar error-correcting codes [13] have been used in recent works [12, 14, 15] to provide data availability, and are commercialized [16]. A recent trend in industry even considers IDA as an alternative to traditional data encryption [17], since IDA provides both data availability and a certain degree of data confidentiality.

1. Related Work In order to support queries on encrypted relational data, one class of solutions proposed processing encrypted data directly. However, these approaches do not provide

good tradeoff between data confidentiality and query efficiency.

2. System and Attacker Model

1 System Model

1.1 Data Model.

1.2 Data Storage Model.

1.3 Data Access Model.

2 Attacker Model

2.1 Attacker and Prior Knowledge Assumptions.

3. Data Encryption and Dispersal by Salted IDA

3.1 Information Dispersal Algorithm (IDA)

We first introduce IDA [12]. IDA encodes and disperses data into n uninterpretable pieces so that only m ($m \le n$) pieces are required to reconstruct the data, and the total storage size of the dispersed pieces is only n/m times of the data size.

3.2 Salted IDA

Based on IDA, we propose a scheme called salted IDA to achieve such data confidentiality. As in IDA, a client maintains an $n \times m$ secret matrix C as the information dispersal matrix and the keys for encoding and decoding a data matrix D, where n,m are determined by the client based on the number of servers that she plans to use and the estimated number of non-faulty servers.

In addition, the client keeps a secret seed ss, and a deterministic function fs for producing random factors based on ss and the address of data entries on D. We call these random factors salt.

4- Secure Cloud Data Access

We use salted IDA to encode and disperse the data onto servers in the cloud. To be able to perform queries on salted IDA encoded matrix, we retrieve partial data by retrieving single columns of the matrix as follows.

$D_{:,i} = C_*^{-1} \cdot E_{*:,i}$

Similarly we can update and encode a single column $D_{:,i}$ as follows.

$E_{:,i} = C \cdot D_{:,i}$ (3)

4.1 Organization of Index

4.2 Organization of Data Tuples

4 .3 Secure Column Access Via Proxies

5-Query Processing

Our framework supports exact, range queries, as well as updates, inserts and deletes. These common queries form the basis for general purpose relational data processing.

6-Security Analysis

In loaded cloud environment with the use of proxies between clients and servers and client side index caches, we ensure data confidentiality against polynomial size circuits bounded attackers, even when all servers are monitored by attackers. We ensure data integrity and availability when no more than n – m servers are faulty.

7-On Data Integrity and Availability

We check integrity violations on the index structure using the relationships of sorted key values and the relationships of nodes in the index, and check integrity violations on data values using the checksums. We rely on IDA to provide data availability when no more than n – m servers are faulty. More details on data integrity and availability can be found in our technical report [23].

8-Experimental Results

8.1 General Overhead Comparison.

8.2 Overhead Breakdown.

8.3 Varying Number of Tuples.

8.4 Varying Query Range/Selectivity.

8.5 Varying Cache Hit Rate.

9-Conclusion

A security analysis and an experimental evaluation indicate our framework achieves a practical trade-off between security and performance.