Executing SQL over Encrypted Data in the Database-Service-Provider Model

The authors propose a new encrypted database framework to prevent data theft at the provider side. The proposed frameworks firstly uses the partition function and identification function to symbolize the numerical value. Mapping function later encrypts the partitioned data. Clients are in charge of both partition, identification and mapping steps and the description is only possible at the client side. On the server side, only encrypted data are store and visible to the service provider. To facilitate standard SQL operations, authors propose standard operators such as equal to, larger than, or condition etc., as well as operations such as table join, sorting etc. The authors also propose heuristics to split queries to process by server and client. Later, experimental results on a benchmark dataset demonstrate the effectiveness of proposed framework, in terms of query processing speed.

Overall, the problem is well motivated, as data privacy is a fundamental problem that start to draw more and more attention. The proposed framework is intuitive and simple. However, I primarily have two concerns.

First, the necessity and the advantage of the partition and identification steps is not well justified. In fact, the proposed framework without the partition and identification is the same as simply encrypt all the fields by mapping functions. Without symbolization (partition and identification) step, the server will be able to return exact result for operations such as join, set different etc., rather than a superset of the exact result under current framework. Furthermore, the split of a query would not be necessary. In terms of privacy concern, a random mapping (non-order preserving) will have the exact effect. It is clear that, for an order preserving mapping function, without symbolization, the provider can infer more information, though not exact. However, this is still true under current framework depending upon the granularity of the partition function, for example, the each partition only covers a small range. There might be advantages by following current approach, such as query execution time. For example, querying records that have value in certain field larger than some threshold could be very time consuming, when non-order preserving mapping functions are used. However, authors fail to demonstrate the advantage through experiment. The author should show comparison between using and not using partition and identification in the experiment, in term of the query speed.

Secondly, the partition step sacrifices the server side utility (e.g., server return a superset of the result). The author fail to show that the utility loss on the server side in under acceptable range. No experiment results show that the client side computation is minimal. The authors should show the ratio of return the records from server being the actual correct records (precision).

What are the possible ways that the service providers could still reveal the encrypted data? For example, through multiply queries, providers might be able to figure out which records have the same attributes value. Eventually, as more and more queries being processed at the service provider side, the provider obtain more information.
Human-powered Sorts and Joins

The authors present an approach for joins and sorts in a crowd sourcing database framework. The proposed system runs on top of crowdsourcing platforms. This paper specifically discusses the implementation of joins and sorts under a crowd-sourcing environment. Two sorting interfaces are explored, i.e., comparison-based and rating-based. One feature filtering technique is proposed to reduce the total number of comparison. The experiment result demonstrate the effectiveness of the system in minimizing the cost when performs sort and join operations.

Overall, this paper is well written and easy to follow. The authors discuss a variety of issues in implementing a working database system on top of a crowdsourcing platform. The authors implement a working system on top of the Mechanical Turk platform.

One important issue that the paper does not elaborate is how to assess the quality of each answer from the crowd. The proposed system employs a majority vote framework. However also looking at the each workers history could provide more reliability.

The proposed join and sort framework generate the question and delivery it to the workers, as the query being processed. The time could be an issue in such framework. The system current does not evaluate the time cost. For time sensitive task, the proposed system might have problems.

For crowd sourcing task, it is often probabilistic answer is good enough rather than compute the exact answer. How approximation algorithm can be integrated in the current sorting scheme?