Two Can Keep a Secret:
A Distributed Architecture for Secure Database Services

Aggarwal et. al.

November 28, 2005
Presented by Siddharth Srivastava
Outline

Motivation

Framework
  • Privacy Model

Methodology
  • Relational Decomposition
  • Example
  • Query Optimization

Choosing Decompositions

Finding Optimal Solutions

Approximate Solutions

Conclusions
Motivation

- Owner has a database with private information about individuals.
- Database needs to be outsourced.
- The Database service provider(s) cannot be trusted.

The entire database is represented as a single relation whose schema is $R$. 
Framework

Client Capability
The client acts as a DBMS front end:

- Reformulates and optimizes queries
- Post-processes query results

Post-processing contributes to extra cost of the solution.

Figure 1: The System Architecture
Framework
Privacy Model

Assumptions

- Servers are honest but curious: record data and queries, but don’t give erroneous results.
- Adversary cannot access both the servers.
- Servers cannot communicate with each other.

Notion of Privacy

- Privacy constraints $\mathcal{P} \subseteq 2^R$
- Breach of privacy: Any element of $\mathcal{P}$ is contained entirely on one server.

This amounts to an adversary not being able to obtain values of all attributes in $p \in \mathcal{P}$ for any tuple.

Example of $p$: $\{\text{DoB}, \text{Gender}, \text{Zip}\}$
Methodology

Relational Decomposition

- Need to decompose the relation $R$ into $R_1, R_2$ for two servers.
- Decomposition should be loss-less and privacy preserving.

Techniques for Relational Decomposition

- Horizontal Decomposition
- Vertical Decomposition
- Semantic Attribute Decomposition
- Attribute Encoding
- Adding Noise
Techniques for Relational Decomposition

Horizontal Decomposition

- Divide tuples between the two servers.
- Maybe useful for statistical queries.

Not very useful for preserving privacy in most settings.

Semantic Attribute Decomposition

- If an attribute has parts that are not private, split the attribute into private and non-private parts.
- Attribute would have to be decomposable on the basis of level of detailed information.
- Similar to anonymizing the attribute value.

We will assume that $R$ is preprocessed to break up all such attributes.
Techniques for Relational Decomposition (ctd.)

### Attribute Encoding

Store the encrypted values on one server and keys on the other.

- **One-time Pad:** \( a_1 = a \oplus r, a_2 = r \).
- **Deterministic Encryption:** \( a_1 = E(a, k), a_2 = k \).
- **Random Addition:** \( a_1 = a + r, a_2 = r \).

**Trade-off:** constant \( r \) implies ease in query optimization (pushing selects), but opens data to statistical attacks.

Attribute Encoding is also viewed as a decomposition: \( a \leftarrow < a_1, a_2 > \)
Techniques for Relational Decomposition (ctd.)

Adding Noise

- Technique for aiding privacy in decomposition.
- Add spurious “dangling” tuples to $R_1$ and $R_2$.
- Any tuple the adversary studies could be a fake.
- Does not help under the current notion of privacy.

Doesn’t help if adversary knows what he/she wants.

Vertical Decomposition

- Separate attributes $a_1, \ldots, a_n$ of $R$ into two sets for the servers.
- Use unique tuple IDs for a loss-less join during retrieval.

Most amenable to the notion of privacy.

Only Vertical Decomposition and Attribute encoding are used in the paper.
Notation

- Input database = Relation \( R = \langle A_1, A_2, \ldots, A_n \rangle \).
- Decomposition of \( R \): \( \langle R_1, R_2, E \rangle \).
- \( E \): set of attributes to be encoded.
- \( E \subseteq R_1 \cap R_2 \).
- \( R = R_1 \cup R_2 \)
Example

Salary Database

\[ R = \{ \text{Name, DoB, Gender, Zip, Position, Salary, Email, Telephone} \} \]
\[ P = \left\{ \{ \text{Telephone, Email} \}, \{ \text{Name, Salary} \}, \{ \text{Name, Position} \}, \{ \text{Name, DoB} \}, \{ \text{Name, DoB} \}, \{ \text{DoB, Gender, Zip} \}, \{ \text{Position, Salary} \}, \{ \text{Salary, DoB} \} \right\} \]

A Possible Decomposition

- \[ R_1 = \{ \text{ID, Name, Gender, Zip, Salary', Email', Telephone'} \} \]
- \[ R_2 = \{ \text{ID, Position, DoB, Salary'', Email'', Telephone''} \} \]
- \[ E = \{ \text{Salary, Email, Telephone} \} \]
Query Optimization

Query optimization involves 3 steps:

▶ Query reformulation: replace $R$ by $R_1 \bowtie R_2$.
▶ Logical query planning.
▶ Physical query planning.

Note: Constructing these plans requires statistics about selectivity on attributes etc., which might have to be constructed via queries.

Logical Query Planning

▶ Construct the sequence in which the query’s sub-operations will be performed so as to minimize computation.
▶ General technique: push down selections, projections and group-by’s as far down as possible before taking joins.
Logical Query Planning: Example

Figure 2: Example of Query Reformulation and Optimization
Query Optimization (ctd.)

Physical Query Planning

- Determine the points at which data exchange takes place between \( S_1 \) and \( S_2 \).
- Can either send queries in parallel and perform a full merge, or send \( S_2 \) the tuple ID’s that will be present in results from \( S_1 \).
- Information Leak? \( S_2 \) already potentially knows all the tuple IDs that are present; \( S_2 \) doesn’t know the query that was sent to \( S_1 \).
Choosing the Best Decomposition

- Trivial solution: encode all.
- Need to find a solution that incurs minimum query-overheads.
- How should we define overheads?

Affinity Matrix

- Idealized view of cost of separation.
- \( M_{ij} = \text{cost of separating } A_i \text{ from } A_j \)
- \( M_{ii} = \text{cost of encoding } A_i \)

This is idealized because cost of separating a pair will depend on the rest of the decomposition.

We use the affinity matrix to define the cost of a decomposition:

\[
\sum_{\{i,j: (A_i \in R_1 \setminus E) \land (A_j \in R_2 \setminus E)\}} M_{ij} + \sum_{i \in E} M_{ii}
\]
Defining the Optimal Solution

The problem of finding the best decomposition thus becomes the following:

**Optimal Decomposition**

Given a set of privacy constraints $\mathcal{P} \subseteq 2^R$, and an affinity matrix $M$, find a decomposition $\mathcal{D}(R) = \langle R_1, R_2, E \rangle$ such that

1. $\mathcal{D}$ obeys all the privacy constraints in $\mathcal{R}$
2. The cost of $\mathcal{D}$ as defined by the affinity matrix is minimized.
Solving the Optimization Problem

Can this problem be solved efficiently?

- Closely related to 2-coloring a graph so as to minimize the weight of bichromatic edges.
- But we also have preferences on the separation.
- Would have been exactly the 2-coloring problem with minimization if constraints were of size 2.
- Need a generalised concept of “edge”.

Hypergraphs

- A hypergraph $H$ is a tuple $\langle V, E \rangle$, where $E \subseteq 2^V$.
- Each edge of a hypergraph is a set of vertices.
Connections with Hypergraph Coloring

- Our problem is the same as 2-coloring a hypergraph $H\langle R, P \rangle$, with the additional constraint that the weight of bi-chromatic graph edges is minimized.

- We also have the possibility of deleting a vertex, but no restriction on the number of elements in each $p \in \mathcal{P}$.

- Coloring a 2-colorable hypergraph where every edge has 4 vertices with $c$ colors is NP-Hard.
Approximate Solutions

- Optimal solution is a Min Cut satisfying the privacy constraints.
- Constraints not satisfied through coloring will be satisfied by attribute encoding.
- Each encoded attribute solves all the privacy constraints containing it.
- Solution = MinCut + Weighted Vertex (Set) Cover.
- Heuristic: find an approximate solution to one problem, starting with an approximate solution of the other.
Approximate Solutions (ctd.)

Approximate Min Cuts

- Ignoring the privacy constraints, we need to find a Min-Cut.
- Finding Cuts within a constant factor of the Minimum-Cut is possible in polynomial time.

Weighted Set-Covers

- Deleting a vertex (or encoding an attribute) from each $p \in \mathcal{P}$ is like finding a set-cover.
- We are interested in the minimum weight set-cover.
- This problem admits a “greedy” $1 + \log |\mathcal{P}|$ approximation.
Heuristic 1

- Find an Approx Wtd Set Cover. Call the set of covering vertices $E$.
- Delete $E$ from $R$.
- Use Approx Min Cuts $C$ to find 2-colorings of $R \setminus E$.
- For each cut $c$ in $C$
  Replace in $R$ all vertices from $E$ which would only be in bichromatic edges.
- Return the best of a replacement solution and the decomposition $\langle R \setminus E, E, E \rangle$. 
Heuristic Algorithms (ctd.)

Heuristic 2
- Find Approx Min Cuts, $C$
- For each cut $c$ in $C$,
  - Delete vertices greedily until all constraints are satisfied.
- Return the best solution among all cuts.

Heuristic 3
Repeat until all constraints are satisfied:
- Start with an Approx Min Cuts solution
- Greedy-select a vertex to delete to satisfy additional constraints.
- Re-run Approx Min Cuts to find cuts without the deleted vertex.
Conclusions

- New Notion of Privacy.
- Connections with distributed databases: possibility of applying developed schemes from there.
- Generalizes $k$-anonymity in a way.
- Provide algorithms for computing the decomposition.
- More analysis is needed to study feasibility of the approach (affinity matrix, algorithms, query planning).
- No running time estimates or experimental results on quality of solutions even with a given affinity matrix. How queries will be degraded in practice is still far from understood.