Processing Analytical Queries over Encrypted Data

Stephen Tu
M. Frans Kaashoek
Sam Madden
Nickolai Zeldovich

VLDB 2013
Introduction

• MONOMI – a system for securely executing *analytical queries* over sensitive data on an untrusted server
  – Builds on top of CryptDB
  – CryptDB can handle only 4/22 TPC-H queries
  – Need to read large fractions of the database but the results are small aggregates
Motivation

• Analytical workloads present 3 main challenges:
  – These queries access a large fraction of the database and are bottlenecked by I/O systems – encryption schemes that significantly increase the size of the data can slow query processing
  – Analytical queries require complex computations – partitioning the query into parts that can be executed using encryption schemes and on the trusted client
  – Techniques for processing queries over encrypted data can speed up certain queries but slow down others – carefully design physical layout and plan query execution
Split Client/Server Execution

- SELECT SUM(price) AS total
  FROM orders
  GROUP BY order_id
  HAVING total > 100

- `order_id` column is encrypted with deterministic encryption – allows the server to find rows that have same values

- `price` column is encrypted with Paillier encryption - allows the server to compute encrypted sum of encrypted values (without decryption key)

- Paillier does not support order comparison – cannot check if total > 100
Split Client/Server Execution

- SELECT PAILLIER_SUM(price_paillier) AS total 
  FROM orders 
  GROUP BY order_id_det

- After the encrypted results of this query are received by the client, it decrypts them and executes HAVING total > 100

<table>
<thead>
<tr>
<th>Encryption scheme</th>
<th>SQL operations</th>
<th>Leakage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Randomized AES + CBC</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Deterministic AES +</td>
<td>a = const, IN,</td>
<td>Duplicates</td>
</tr>
<tr>
<td>OPE [6]</td>
<td>a &gt; const, MAX, ORDER BY</td>
<td>Order + partial</td>
</tr>
<tr>
<td>Paillier [20]</td>
<td>a + b, SUM(a)</td>
<td>plaintext [7]</td>
</tr>
<tr>
<td>SEARCH [22, 24]</td>
<td>a LIKE pattern</td>
<td>None</td>
</tr>
</tbody>
</table>
Optimization Techniques

• PER-ROW PRECOMPUTATION
  – SUM(ps_supplycost*ps_availaqty)
  – EXTRACT(YEAR FROM o_orderdate)
• Paillier encryption cannot perform multiplication of two values
• YEAR cannot be computed without making the encryption schemes less secure and less space efficient
• Materialize additional column in table containing encrypted value of an expression e.g. store the homomorphic encryption of ps_supplycost*ps_availqty – allows the server to compute the SUM
• Also allows the client to download one encrypted value instead of both values
Optimization Techniques

• SPACE-EFFICIENT ENCRYPTION
  – Table scans are common analytic queries and are bottlenecked by I/O
  – Performance of such queries are sensitive to encrypted on-disk data
  – FFX block cipher mode of operation: maps n-bit plaintexts to n-bit ciphertexts, given $n < \text{block cipher width}$ (128 bits for AES)
  – For values longer than block cipher width CTS mode combined with CMC mode provides an alternative
Optimization Techniques

• GROUPED HOMOMORPHIC ADDITION
  – Paillier homomorphic scheme: \( E(a + b) = E(a) \times E(b) \)
  – Modular multiplications are expensive – need to perform modulo 2048-bit public key
  – Plaintext for row \( a \): \((a_1 || a_2 || \ldots || a_k)\)
  – \( E(a_1 || a_2 || \ldots || a_k) \times E(b_1 || b_2 || \ldots || b_k) = E((a_1 + b_1) || (a_2 + b_2) || \ldots || (a_k + b_k)) \)
  – Can be done with one modular multiplication per row
  – Given the product of all row ciphertexts, client can extract the aggregate of column \( i \) from position \( I \) in the decrypted ciphertext
Optimization Techniques

• CONSERVATIVE PRE-FILTERING

```sql
SELECT l_orderkey FROM lineitem
GROUP BY l_orderkey
HAVING SUM(l_quantity) > :1

SELECT l_orderkey_det,
    PAILLIER_SUM(l_quantity_paillier)
FROM lineitem
GROUP BY l_orderkey_det
HAVING MAX(l_quantity_ope) > encrypt_ope(m)
    OR COUNT(*) > (:1 / m)
```
Physical Database Design

• Physical DB designer: chooses the best set of
  – expressions to precompute
  – Encryption schemes to create for each column

• Designer uses a query planner to choose the best query execution strategy given a design

• To determine the physical design, user provides a query workload and a sample of data
Physical Database Design

• For each operation in Qi, the designer determines what expression and encryption scheme would allow the operation to execute on the server
• The designer invokes the planner to determine how to best execute Qi given the encryption schemes
• For each of the execution plans, the planner uses a cost model to determine the best execution plan for Qi
• The designer takes the union of the encryption schemes required by each query and uses that as the physical design
Thanks!

Questions ???