Secure Skyline Queries on Encrypted Data

CS 573 Data Privacy and Security


2018-11-19
Skyline Computation: Hotel Example

<table>
<thead>
<tr>
<th>hotel</th>
<th>distance</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>4</td>
<td>400</td>
</tr>
<tr>
<td>p2</td>
<td>24</td>
<td>380</td>
</tr>
<tr>
<td>p3</td>
<td>14</td>
<td>340</td>
</tr>
<tr>
<td>p4</td>
<td>36</td>
<td>300</td>
</tr>
<tr>
<td>p5</td>
<td>26</td>
<td>280</td>
</tr>
<tr>
<td>p6</td>
<td>8</td>
<td>260</td>
</tr>
<tr>
<td>p7</td>
<td>40</td>
<td>200</td>
</tr>
<tr>
<td>p8</td>
<td>20</td>
<td>180</td>
</tr>
<tr>
<td>p9</td>
<td>34</td>
<td>140</td>
</tr>
<tr>
<td>p10</td>
<td>28</td>
<td>120</td>
</tr>
<tr>
<td>p11</td>
<td>16</td>
<td>60</td>
</tr>
</tbody>
</table>

Secure Skyline Queries on Encrypted Data
Skyline Computation: Hotel Example

<table>
<thead>
<tr>
<th>hotel</th>
<th>distance</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>p1</td>
<td>4</td>
<td>400</td>
</tr>
<tr>
<td>p2</td>
<td>24</td>
<td>380</td>
</tr>
<tr>
<td>p3</td>
<td>14</td>
<td>340</td>
</tr>
<tr>
<td>p4</td>
<td>36</td>
<td>300</td>
</tr>
<tr>
<td>p5</td>
<td>26</td>
<td>280</td>
</tr>
<tr>
<td>p6</td>
<td>8</td>
<td>260</td>
</tr>
<tr>
<td>p7</td>
<td>40</td>
<td>200</td>
</tr>
<tr>
<td>p8</td>
<td>20</td>
<td>180</td>
</tr>
<tr>
<td>p9</td>
<td>34</td>
<td>140</td>
</tr>
<tr>
<td>p10</td>
<td>28</td>
<td>120</td>
</tr>
<tr>
<td>p11</td>
<td>16</td>
<td>60</td>
</tr>
</tbody>
</table>
Motivating Example: Skyline Queries

Table: Sample of heart disease dataset.

(a) Original data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁</td>
<td>40</td>
<td>140</td>
</tr>
<tr>
<td>p₂</td>
<td>39</td>
<td>120</td>
</tr>
<tr>
<td>p₃</td>
<td>45</td>
<td>130</td>
</tr>
<tr>
<td>p₄</td>
<td>37</td>
<td>140</td>
</tr>
</tbody>
</table>

(b) Mapped Data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₃</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t₄</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure: q(41,125).
Motivating Example: Skyline Queries

Table: Sample of heart disease dataset.

(a) Original data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁</td>
<td>40</td>
<td>140</td>
</tr>
<tr>
<td>p₂</td>
<td>39</td>
<td>120</td>
</tr>
<tr>
<td>p₃</td>
<td>45</td>
<td>130</td>
</tr>
<tr>
<td>p₄</td>
<td>37</td>
<td>140</td>
</tr>
</tbody>
</table>

(b) Mapped Data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>42</td>
<td>140</td>
</tr>
<tr>
<td>t₂</td>
<td>43</td>
<td>130</td>
</tr>
<tr>
<td>t₃</td>
<td>45</td>
<td>130</td>
</tr>
<tr>
<td>t₄</td>
<td>45</td>
<td>140</td>
</tr>
</tbody>
</table>

Figure: q(41,125).
Motivating Example: Skyline Queries

Table: Sample of heart disease dataset.

(a) Original data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>p₁</td>
<td>40</td>
<td>140</td>
</tr>
<tr>
<td>p₂</td>
<td>39</td>
<td>120</td>
</tr>
<tr>
<td>p₃</td>
<td>45</td>
<td>130</td>
</tr>
<tr>
<td>p₄</td>
<td>37</td>
<td>140</td>
</tr>
</tbody>
</table>

(b) Mapped Data.

<table>
<thead>
<tr>
<th>ID</th>
<th>age</th>
<th>trestbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>42</td>
<td>140</td>
</tr>
<tr>
<td>t₂</td>
<td>43</td>
<td>130</td>
</tr>
<tr>
<td>t₃</td>
<td>45</td>
<td>130</td>
</tr>
<tr>
<td>t₄</td>
<td>45</td>
<td>140</td>
</tr>
</tbody>
</table>

Figure: q(41,125).
Secure Similarity Queries
Related Work

- Fully homomorphic encryption - impractical
Related Work

- Fully homomorphic encryption - impractical
- Order preserving encryption - subjective to attacks
Related Work

- Fully homomorphic encryption - impractical
- Order preserving encryption - subjective to attacks
- Partially homomorphic encryption - limited computation but efficient, many focused on knn queries, challenging for skyline due to complex comparisons
Problem setting
Outline

- Problem setting
- Paillier crypto scheme

Secure Skyline Queries on Encrypted Data
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
- Secure skyline protocol
Problem setting
Paillier crypto scheme
Basic primitive subprotocols
Secure dominance protocol
Secure skyline protocol
Experimental results
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
- Secure skyline protocol
- Experimental results
Problem Setting

Client: q, pk

C1: $E_{pk}(P), E_{pk}(q), pk$

Data owner: P, pk, sk

$E_{pk}(q)$

skyline result

$E_{pk}(P)$
Problem Setting

Client: $q, pk$

$C_1: E_{pk}(P), E_{pk}(q), pk$

Data owner: $P, pk, sk$

Secure Skyline Queries on Encrypted Data
Problem Setting

Client:
q, pk

C1:
E_{pk}(P), E_{pk}(q), pk

Data owner:
P, pk, sk

C2:
partial skyline result

sk, partial skyline result

C1 and C2 are non-colluding
Problem Setting

Data owner (e.g., hospital, CDC) sends private key to $C_2$.
Data owner sends $E_{pk}(p_i[j])$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$ to cloud server $C_1$. 
An authorized client (e.g., physician) sends $E_{pk}(q)$ to cloud server $C_1$. 
Problem Setting

Our goal is to enable the cloud server to **compute** and return the skyline to the client without learning any information about the data and the query.
Problem Setting

Our goal is to enable the cloud server to compute and return the skyline to the client without learning any information about the data and the query.
Data Privacy. Cloud servers $C_1$ and $C_2$ know nothing about the exact data except the size pattern, the client knows nothing about the dataset except the skyline query result.
Data Privacy. Cloud servers $C_1$ and $C_2$ know nothing about the exact data except the size pattern, the client knows nothing about the dataset except the skyline query result.
Data Pattern Privacy. Cloud servers $C_1$ and $C_2$ know nothing about the data patterns (indirect data knowledge) due to intermediate result, e.g., which tuple dominates which other tuple.
Problem Setting: Desired Privacy Properties

Client: $q, pk$

Data owner: $P, pk, sk$

$E_{pk}(q), E_{pk}(P), pk$

$C_1:$

$C_2:$

Query Privacy. Data owner, cloud servers $C_1$ and $C_2$ know nothing about the query tuple $q$. 
Problem Setting: Desired Privacy Properties

Result Privacy. Cloud servers $C_1$ and $C_2$ know nothing about the query result, e.g., which tuples are in the skyline result.
Problem setting
Paillier crypto scheme
Basic primitive subprotocols
Secure dominance protocol
Secure skyline protocol
Experimental results
• Homomorphic addition of plaintexts:

\[ D_{sk}(E_{pk}(a) \times E_{pk}(b) \mod N^2) = (a + b) \mod N \]

• Homomorphic multiplication of plaintexts:

\[ D_{sk}(E_{pk}(a)^b \mod N^2) = a \times b \mod N \]

https://mhe.github.io/jspaillier/
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
- Secure skyline protocol
- Experimental results
Basic Security Subprotocols: Secure Multiplication (SM)

- **Input**
  - $C_1$: encrypted input $E_{pk}(a)$ and $E_{pk}(b)$
  - $C_2$: private key $sk$

- **Output**
  - $C_1$ knows $E_{pk}(a \times b)$
  - $C_2$ knows nothing
Basic Security Subprotocols: Secure Bit Decomposition (SBD)

**Input**
- $C_1$: encrypted input $E_{pk}(a)$
- $C_2$: private key $sk$

**Output**
- $C_1$ knows encrypted individual bits of the binary representation of $a$, denoted as $\llbracket a \rrbracket = \langle E_{pk}((a)_B^{(1)}), \ldots, E_{pk}((a)_B^{(l)}) \rangle$, where $l$ is the number of bits, $(a)_B^{(1)}$ and $(a)_B^{(l)}$ denote the most and least significant bits of $a$, respectively.
- $C_2$ knows nothing.
Basic Security Subprotocols

- Secure OR (SOR)
- Secure AND (SAND)
- Secure NOT (SNOT)
- Secure Less Than or Equal (SLEQ)
- Secure Equal (SEQ)
- Secure Less (SLESS)
- Secure Minimum (SMIN)
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- **Secure dominance protocol**
- Secure skyline protocol
- Experimental results
For each comparison between two tuples $p_a$ and $p_b$, we need to compare all their $m$ attributes and for comparison of each attribute $p[j]$, there are three different outputs, i.e., $p_a[j] < (\leq, >) p_b[j]$. Therefore, there are $3^m$ different outputs for each comparison between two tuples, based on which we need to determine if one tuple dominates the other.
For each comparison between two tuples $p_a$ and $p_b$, we need to compare all their $m$ attributes and for comparison of each attribute $p[j]$, there are three different outputs, i.e., $p_a[j] < (\leq, >) p_b[j]$.

Therefore, there are $3^m$ different outputs for each comparison between two tuples, based on which we need to determine if one tuple dominates the other.
Algorithm 1 Secure Dominance Protocol.

1: **Input:** $C_1$ has $E_{pk}(a)$, $E_{pk}(b)$ and $C_2$ has $sk$.
2: **Output:** $C_1$ gets $E_{pk}(1)$ if $a \prec b$, otherwise, $C_1$ gets $E_{pk}(0)$.
3: $C_1$ and $C_2$:
4: for $j = 1$ to $m$ do
5: $C_1$ gets $\delta_j = E_{pk}(\text{Bool}(a[j] \leq b[j]))$ by SLEQ
6: end for
7: use SAND to compute $\Phi = \delta_1 \land \ldots, \land \delta_m$
8: $C_1$:
9: compute $\alpha = E_{pk}(a[1]) \times \ldots, \times E_{pk}(a[m])$
10: compute $\beta = E_{pk}(b[1]) \times \ldots, \times E_{pk}(b[m])$
11: $C_1$ and $C_2$:
12: $C_1$ gets $\sigma = E_{pk}(\text{Bool}(\alpha < \beta))$ by employing SLESS
13: $C_1$ gets $\Psi = \sigma \land \Phi$ as the final dominance relationship using SAND

CS 573 Data Privacy and Security | Secure Skyline Queries on Encrypted Data
Algorithm 2 Secure Dominance Protocol.

1: Input: $C_1$ has $E_{pk}(a)$, $E_{pk}(b)$ and $C_2$ has sk. $a=(2,5); b=(4,5)$
2: Output: $C_1$ gets $E_{pk}(1)$ if $a \prec b$, otherwise, $C_1$ gets $E_{pk}(0)$.
3: $C_1$ and $C_2$:
4: for $j = 1$ to $m$ do
5: $C_1$ gets $\delta_j = E_{pk}(\text{Bool}(a[j] \leq b[j]))$ by SLEQ $\delta_1 = 1; \delta_2 = 1$
6: end for
7: use SAND to compute $\Phi = \delta_1 \land \ldots \land \delta_m \Phi = 1$
8: $C_1$:
9: compute $\alpha = E_{pk}(a[1]) \times \ldots \times E_{pk}(a[m]) \alpha = 7$
10: compute $\beta = E_{pk}(b[1]) \times \ldots \times E_{pk}(b[m]) \beta = 9$
11: $C_1$ and $C_2$:
12: $C_1$ gets $\sigma = E_{pk}(\text{Bool}(\alpha < \beta))$ by employing SLESS $\sigma = 1$
13: $C_1$ gets $\Psi = \sigma \land \Phi$ as the final dominance relationship using SAND $\Psi = 1$
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
- Secure skyline protocol
- Experimental results
Algorithm 3 Skyline Computation.

1: **Input:** A dataset $T$.
2: **Output:** Skyline of $T$.
3: **while** the dataset $T$ is not empty **do**
4:  
5:  
6: choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline
7: add $t_{\text{min}}$ to skyline pool
8: delete those tuples dominated by $t_{\text{min}}$ from $T$
9: delete tuple $t_{\text{min}}$ from $T$
10: **end for**
11: **end while**
12: **return** skyline pool
Algorithm 4 Skyline Computation.

1: **Input:** A dataset $T$.
2: **Output:** Skyline of $T$.
3: while the dataset $T$ is not empty do
4:     for $i = 1$ to size of dataset $T$ do
5:         $S(t_i) = \sum_{j=1}^{m} t_i[j]$  
6:             choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline
7:             add $t_{\text{min}}$ to skyline pool
8:             delete those tuples dominated by $t_{\text{min}}$ from $T$
9:             delete tuple $t_{\text{min}}$ from $T$
10:    end for
11: end while
12: return skyline pool
Algorithm 5 Skyline Computation.

1: Input: A dataset $T$.
2: Output: Skyline of $T$.
3: while the dataset $T$ is not empty do
4:     for $i = 1$ to size of dataset $T$ do
5:         $S(t_i) = \sum_{j=1}^{m} t_i[j]$
6:         choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline
7:         add $t_{\text{min}}$ to skyline pool
8:         delete those tuples dominated by $t_{\text{min}}$ from $T$
9:         delete tuple $t_{\text{min}}$ from $T$
10:     end for
11: end while
12: return skyline pool

Skyline Pool

$t_1$ $t_2$ $t_3$ $t_4$
Algorithm 6 Skyline Computation.

1: Input: A dataset $T$.
2: Output: Skyline of $T$.
3: while the dataset $T$ is not empty do
4:     for $i = 1$ to size of dataset $T$ do
5:         $S(t_i) = \sum_{j=1}^{m} t_i[j]$
6:         choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline
7:         add $t_{min}$ to skyline pool
8:         delete those tuples dominated by $t_{min}$ from $T$
9:         delete tuple $t_{min}$ from $T$
10:     end for
11: end while
12: return skyline pool

Skyline Pool

$t_1$ $t_2$ $t_3$ $t_4$
Algorithm 7 Skyline Computation.

1: Input: A dataset $T$.
2: Output: Skyline of $T$.
3: while the dataset $T$ is not empty do
4:   for $i = 1$ to size of dataset $T$ do
5:     $S(t_i) = \sum_{j=1}^{m} t_{ij}$
6:     choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline
7:     add $t_{\text{min}}$ to skyline pool
8:     delete those tuples dominated by $t_{\text{min}}$ from $T$
9:     delete tuple $t_{\text{min}}$ from $T$
10: end for
11: end while
12: return skyline pool

Skyline Pool

$t_1$
$t_2$

$15$
$10$
$5$

$1$
$2$
$3$
$4$
**Algorithm 8 Skyline Computation.**

1: **Input**: A dataset $T$.
2: **Output**: Skyline of $T$.
3: while the dataset $T$ is not empty do
4: for $i = 1$ to size of dataset $T$ do
5: $S(t_i) = \sum_{j=1}^m t_{i[j]}$
6: choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline
7: add $t_{\text{min}}$ to skyline pool
8: delete those tuples dominated by $t_{\text{min}}$ from $T$
9: delete tuple $t_{\text{min}}$ from $T$
10: end for
11: end while
12: return skyline pool
**Algorithm 9** Skyline Computation.

1: **Input:** A dataset $T$.
2: **Output:** Skyline of $T$.
3: while the dataset $T$ is not empty do
4:      for $i = 1$ to size of dataset $T$ do
5:         $S(t_i) = \sum_{j=1}^{m} t_i[j]$
6:         choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline
7:         add $t_{min}$ to skyline pool
8:         delete those tuples dominated by $t_{min}$ from $T$
9:         delete tuple $t_{min}$ from $T$
10:      end for
11:  end while
12:  return skyline pool

![Skyline Computation Algorithm](image)
choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline

*Initial case $t_i$*

\[
\begin{array}{|c|c|}
\hline
\text{Party $C_1$} & \text{C}_2 \\
\hline
\text{t}_i & (t_i[1], t_i[2]) \\
\hline
\text{t}_1 & (1, 15) \\
\hline
\text{t}_2 & (2, 5) \\
\hline
\text{t}_3 & (4, 5) \\
\hline
\text{t}_4 & (4, 15) \\
\hline
\end{array}
\]
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

$$E_{pk}(S(t_i)) = E_{pk}(t_i[1]) \times \ldots \times E_{pk}(t_i[m]) \mod N^2$$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>$(1, 15)$</td>
<td>16</td>
</tr>
<tr>
<td>$t_2$</td>
<td>$(2, 5)$</td>
<td>7</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$(4, 5)$</td>
<td>9</td>
</tr>
<tr>
<td>$t_4$</td>
<td>$(4, 15)$</td>
<td>19</td>
</tr>
</tbody>
</table>
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

$$[[E_{pk}(S(t_i))]] = SBD(E_{pk}(S(t_i)))$$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
</tr>
</tbody>
</table>
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

*add a $\lceil \log n \rceil$ − bit sequence to the end of each $E_{pk}(S(t_i))$*

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>$pert.$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
</tr>
</tbody>
</table>
choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline

*perturbed values guaranteed to be different while order is preserved*

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>pert.</th>
<th>$S(t_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
</tr>
</tbody>
</table>
choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline

finding smallest $S(t_i)$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>$\text{pert.}$</th>
<th>$S(t_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
</tr>
</tbody>
</table>
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

$$E_{pk}(S(t_{min}))^{N-1} \times E_{pk}(S(t_i)) \mod N^2$$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>pert.</th>
<th>$S(t_i)$</th>
<th>$S(t_i) - S(t_{min})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
<td>67 − 30</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
<td>30 − 30</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
<td>37 − 30</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
<td>76 − 30</td>
</tr>
</tbody>
</table>
Secure Skyline Protocol

choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

*randomly noise vector* $r$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>pert.</th>
<th>$S(t_i)$</th>
<th>$S(t_i) - S(t_{min})$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
<td>67 − 30</td>
<td>3</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
<td>30 − 30</td>
<td>9</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
<td>37 − 30</td>
<td>31</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
<td>76 − 30</td>
<td>2</td>
</tr>
</tbody>
</table>
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

*permutation sequence* $\pi$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>($t_i[1], t_i[2]$)</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>pert.</th>
<th>$S(t_i)$</th>
<th>$S(t_i) - S(t_{min})$</th>
<th>$r$</th>
<th>$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
<td>67 − 30</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
<td>30 − 30</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
<td>37 − 30</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
<td>76 − 30</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
choose the tuple $t_{min}$ with smallest $S(t_i)$ as a skyline

$$\pi(E_{pk}(S(t_{min}))^{N-1} \times E_{pk}(S(t_i)))^{r_i}$$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$S(t_i)$</th>
<th>$[[S(t_i)]]$</th>
<th>pert.</th>
<th>$S(t_i)$</th>
<th>$S(t_i) - S(t_{min})$</th>
<th>$r$</th>
<th>$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>16</td>
<td>1, 0, 0, 0, 0</td>
<td>1, 1</td>
<td>67</td>
<td>67 − 30</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>7</td>
<td>0, 0, 1, 1, 1</td>
<td>1, 0</td>
<td>30</td>
<td>30 − 30</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>9</td>
<td>0, 1, 0, 0, 1</td>
<td>0, 1</td>
<td>37</td>
<td>37 − 30</td>
<td>31</td>
<td>4</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>19</td>
<td>1, 0, 0, 1, 1</td>
<td>0, 0</td>
<td>76</td>
<td>76 − 30</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

$\beta'$

<table>
<thead>
<tr>
<th>$C_2$</th>
<th>$eta'$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>111</td>
</tr>
<tr>
<td></td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>217</td>
</tr>
</tbody>
</table>
choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline

$$\text{if } \beta'_i = 0, U_i = E_{pk}(1)$$
choose the tuple $t_{\text{min}}$ with smallest $S(t_i)$ as a skyline

$$V = \pi'(U)$$

<table>
<thead>
<tr>
<th>$t_i$</th>
<th>$(t_i[1], t_i[2])$</th>
<th>$V$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
<td>0</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
<td>0</td>
</tr>
</tbody>
</table>
add skyline tuple to skyline pool

\[ \mathbf{t}_i'[j] = V_i \times \mathbf{t}_i[j] \]

<table>
<thead>
<tr>
<th>( \mathbf{t}_i )</th>
<th>( (\mathbf{t}_i[1], \mathbf{t}_i[2]) )</th>
<th>( V )</th>
<th>( (\mathbf{t}_i[1]', \mathbf{t}_i[2]') )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{t}_1 )</td>
<td>(1, 15)</td>
<td>0</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>( \mathbf{t}_2 )</td>
<td>(2, 5)</td>
<td>1</td>
<td>(2, 5)</td>
</tr>
<tr>
<td>( \mathbf{t}_3 )</td>
<td>(4, 5)</td>
<td>0</td>
<td>(0, 0)</td>
</tr>
<tr>
<td>( \mathbf{t}_4 )</td>
<td>(4, 15)</td>
<td>0</td>
<td>(0, 0)</td>
</tr>
</tbody>
</table>
Secure Skyline Protocol

add skyline tuple to skyline pool

\[ p_i'[j] = V_i \times p_i[j] \]

<table>
<thead>
<tr>
<th>( t_i )</th>
<th>((t_i[1], t_i[2]))</th>
<th>( V )</th>
<th>((t_i[1]', t_i[2]'))</th>
<th>((p_i[1]', p_i[2]'))</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>((1, 15))</td>
<td>0</td>
<td>((0, 0))</td>
<td>((0, 0))</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>((2, 5))</td>
<td>1</td>
<td>((2, 5))</td>
<td>((39, 120))</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>((4, 5))</td>
<td>0</td>
<td>((0, 0))</td>
<td>((0, 0))</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>((4, 15))</td>
<td>0</td>
<td>((0, 0))</td>
<td>((0, 0))</td>
</tr>
</tbody>
</table>
eliminate non-skyline tuples

\[ C_1 \text{ and } C_2 \text{ use SOR with } V \text{ to make } E_{pk}(S(t_{\min})) = E_{pk}(127) \]

<table>
<thead>
<tr>
<th>Party C_1</th>
<th>C_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>t_i</td>
<td>(t_i[1], t_i[2])</td>
</tr>
<tr>
<td>t_1</td>
<td>(1, 15)</td>
</tr>
<tr>
<td>t_2</td>
<td>(2, 5)</td>
</tr>
<tr>
<td>t_3</td>
<td>(4, 5)</td>
</tr>
<tr>
<td>t_4</td>
<td>(4, 15)</td>
</tr>
</tbody>
</table>
**Secure Skyline Protocol**

eliminate non-skyline tuples

**secure dominance protocol**

<table>
<thead>
<tr>
<th><strong>Party C₁</strong></th>
<th><strong>C₂</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_i )</td>
<td>( (t_i[1], t_i[2]) )</td>
</tr>
<tr>
<td>( t_1 )</td>
<td>( (1, 15) )</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>( (2, 5) )</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>( (4, 5) )</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>( (4, 15) )</td>
</tr>
</tbody>
</table>
Secure Skyline Protocol

eliminate non-skyline tuples

make $E_{pk}(S(t_i)) = E_{pk}(127)$, where $t_i$ is dominated by $t_{min}$

<table>
<thead>
<tr>
<th>Party $C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_i$</td>
<td></td>
</tr>
<tr>
<td>$t_1$</td>
<td>(1, 15)</td>
</tr>
<tr>
<td>$t_2$</td>
<td>(2, 5)</td>
</tr>
<tr>
<td>$t_3$</td>
<td>(4, 5)</td>
</tr>
<tr>
<td>$t_4$</td>
<td>(4, 15)</td>
</tr>
</tbody>
</table>
Outline

- Problem setting
- Paillier crypto scheme
- Basic primitive subprotocols
- Secure dominance protocol
- Secure skyline protocol
- Experimental results
Experiment Setup

Protocols:
- BSSP: Basic Secure Skyline Protocol
- FSSP: Fully Secure Skyline Protocol

Datasets:
- NBA: real NBA dataset
- INDE: independent dataset
- CORR: correlated dataset
- ANTI: anti-correlated dataset

Goal:
- evaluate the performance and scalability of our protocols
The impact of $n$ ($m=2$, $K=512$)

(a) time cost of CORR

(b) time cost of INDE

(c) time cost of ANTI

(d) time cost of NBA

Secure Skyline Queries on Encrypted Data
The impact of $m$ ($n=1000$, $K=512$)

(e) time cost of CORR

(f) time cost of INDE

(g) time cost of ANTI

(h) time cost of NBA
The impact of K (n=1000, m=2)

(i) time cost of CORR

(j) time cost of INDE

(k) time cost of ANTI

(l) time cost of NBA
Conclusion

- Proposed a secure dominance sub-protocol.
- Proposed a fully secure skyline protocol.
- Demonstrated practical using simulation.

Future work

- Further optimization of algorithm complexity and running time.
Thank You!!!