In the news...

Google's Technique: How It Worked

The Internet giant circumvented privacy settings on Apple's Safari browser.

Safari automatically prevents installation of 'cookies'—small files that can track a person's Web browsing—from all networks and other so-called third parties.

Google, until recently assured Safari users on one of its sites that because of this, they don't need to opt out of Google tracking:

However, Google exploited a loophole in Safari. It allows an advertiser to place a cookie if the user interacts with the ad.

Some ads placed by DoubleClick (which Google owns) made it appear to Safari that the user was purposely interacting with DoubleClick by automatically sending an invisible form.

Safari would thus allow DoubleClick to install a temporary cookie on the user's computer.

After that, the user's browsing activity could, in many cases, be tracked widely across the Web.
Database privacy

- Two general methods to deal with database privacy
  - Query restriction: Limit what queries are allowed. Allowed queried are answered correctly, while disallowed queries are simply not answered
  - Perturbation: Queries answered “noisily”. Also includes “scrubbing” (or suppressing) some of the data

- (Could also be combined)

Query restriction

- Basic form of query restriction: only allow queries that involve more than some threshold $t$ of users

- Example: only allow sum/average queries about a set $S$ of people, where $|S| \geq 5$ (say)
### Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>F</td>
<td>12</td>
<td>$65,000</td>
</tr>
<tr>
<td>Bob</td>
<td>M</td>
<td>1</td>
<td>$40,000</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>20</td>
<td>$70,000</td>
</tr>
<tr>
<td>Dan</td>
<td>M</td>
<td>30</td>
<td>$80,000</td>
</tr>
<tr>
<td>Evan</td>
<td>M</td>
<td>4</td>
<td>$50,000</td>
</tr>
<tr>
<td>Frank</td>
<td>M</td>
<td>8</td>
<td>$58,000</td>
</tr>
</tbody>
</table>

Give me SUM Salary WHERE Gender='F'

Request denied!

+ **Query restriction**
  - Basic query restriction doesn't work…
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>F</td>
<td>12</td>
<td>$65,000</td>
</tr>
<tr>
<td>Bob</td>
<td>M</td>
<td>1</td>
<td>$40,000</td>
</tr>
<tr>
<td>Charlie</td>
<td>M</td>
<td>20</td>
<td>$70,000</td>
</tr>
<tr>
<td>Dan</td>
<td>M</td>
<td>30</td>
<td>$80,000</td>
</tr>
<tr>
<td>Evan</td>
<td>M</td>
<td>4</td>
<td>$50,000</td>
</tr>
<tr>
<td>Frank</td>
<td>M</td>
<td>8</td>
<td>$58,000</td>
</tr>
</tbody>
</table>

Give me SUM Salary WHERE Gender='M'

Evan's salary: $50,000
Frank's salary: $58,000

$363,000 $298,000

Note

- Each query on its own is allowed
- But inference becomes possible once both queries are made
Basic query restriction

- Basic query restriction alone doesn't work when multiple queries are allowed

- Similar problems arise if the database is dynamic
  - E.g., determine a person's salary after they are hired by making the same query (over the entire database) before and after their hire date

Query restriction

- Can use more complicated forms of query restriction based on all prior history
  - E.g., if query for S was asked, do not allow query for a set S' if |S'ΔS| is “small”

- Drawbacks
  - Maintaining the entire query history is expensive
  - Difficult to specify what constitutes a privacy “breach”
  - Does not address adversary’s external information
Query restriction

- Comparing queries pairwise is not enough!

- Example
  - Say you want information about user i
  - Let S, T be non-overlapping sets, not containing i
  - Ask for \( \text{SUM}(\text{Salary, } S) \), \( \text{SUM}(\text{salary, } T) \), and \( \text{SUM}(\text{salary, } S \cup T \cup \{i\}) \)

- Inference can be very difficult to detect and prevent…
  - NP-complete (in general) to determine whether a breach has occurred

Query restriction

- Apply query restriction across all users, or on a per-user basis?
  - If the former, usability limited
  - If the latter, security can be compromised by colluding users
Query restriction

- Query restriction itself may reveal information!

- Example: say averages released only if there are at least 2 data points being averaged
  - Request the average salary of all employees whose GPA is \( \geq X \)
  - No response means that there are fewer than 2 employees with GPA \( \geq X \)
  - If query(GPA \( \geq X \)) answered but query(GPA \( \geq X + \Delta \)) is not, there is at least one employee whose GPA lies between \( X \) and \( X + \Delta \)

- Another example: say we don't want an adversary to learn our exact age
  - Deny query if the answer would exactly reveal the age

- Say age=30
  - Adversary asks “is age \( \geq 30 \)?”, gets response “yes”
  - Adversary asks “is age \( \leq 30 \)?”
    - Correct answer reveals the exact age!
    - But denying the query reveals the exact age also...
Query restriction

- Another example: say we do not want an adversary to learn any value $x, y, z$ exactly

- Consider the table with $x = y = z = 1$, where it is known that $x, y, z \in \{0, 1, 2\}$

- User requests $\text{sum}(x, y, z)$, gets response 3

- User requests $\text{max}(x, y, z)$
  - If user learns the answer, can deduce that $x = y = z = 1$
  - But if the request is denied, the user can still deduce that $x = y = z = 1$ (!!)

Query restriction

- We can try to “look ahead”, and not respond to any query for which there is a subsequent query that will reveal information regardless of whether we respond or not

```
  deny  sum(x, y, z)
  /     \
respond? / \
  max(x, y, z)
  /     \
respond?  deny?
```
Query restriction with “look-aheads”

- Problems
  - May need to look more than 1 level deep
  - Computationally infeasible, even if only looking 1 level deep
  - Does it even work?
    - Denying "is age ≥ 30?" reveals that age=30
    - Denying the request for sum(x, y, z) reveals that x = y = z

- Even if answers don't uniquely reveal a value, they may leak lots of partial information

Query restriction

- A different approach: “simulatable auditing”

- Deny query if there is some database for which that query would reveal information

- This fixes the previous problems

- Even more computationally expensive

- Restricts usability – most queries denied
Electronic Cash