Finished with Authentication

Moving into Privacy

Privacy and anonymity

• Database privacy
• [Privacy in social networks]
• [Privacy on the web]
• [Anonymous communication]
• [Pseudonymity]

• None of these are addressed by any of the techniques we have discussed so far!
What is different here?

- Privacy/pseudonymity
  - Different trust relationships – interactions between entities that partially trust each other
  - Inherently “fuzzy” – ok if a few people learn some information; not ok if everyone learns all information

- Anonymity
  - Classical crypto hides the contents of what is being communicated, but not the fact that communication is taking place

Databases...

- Several possibilities
  - Medical data
  - Scientific research (on human subjects)
  - US census data
  - Employment data
  - ...
  - Data about oneself (e.g., on smartphone)
Two models

• Non-interactive data disclosure
  • Users given access to “all data” (after the data is anonymized/sanitized/processed in some way)
  • Note: it does not suffice to just delete the names!

• Interactive mechanisms
  • Users given the ability to query the database

Database privacy

• Want to be able to discern statistical trends without violating (individual) privacy
  • An inherent tension!

• Questions:
  • [How to obtain the raw data in the first place?]
  • How to allow effective data mining while still maintaining (some level of) user privacy?

• Serious real-world problem
  • Federal laws regarding medical privacy
  • Data mining on credit card transactions, web browsing, movie recommendations, …
Database privacy

• A user (or group of users) has authorized access to certain data in a database, but not to all data
  • E.g., user is allowed to learn certain entries only
  • E.g., user is allowed to learn aggregate data but not individual data (e.g., allowed to learn the average salary but not individual salaries)
  • E.g., allowed to learn trends (i.e., data mining) but not individual data

• How to enforce?

• Note: we are assuming that authentication/access control is already taken care of...

Database privacy

• The problem is compounded by the fact that 'allowing effective data mining' and 'privacy' are (usually) left vague
  • If so, solutions are inherently heuristic and ad-hoc

• Recent work toward formally pinning down what these notions mean
The problem

• A user may be able to learn unauthorized information via inference
  • Combining multiple pieces of authorized data
  • Combining authorized data with “external” knowledge
    • 87% of people identified by ZIP code + gender + date of birth
    • Someone with breast cancer is likely a female

• This is a (potentially) serious real-world problem
  • See the article by Sweeney for many examples

Example

• Say not allowed to learn any individual's salary

<table>
<thead>
<tr>
<th>Name</th>
<th>UID</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>001</td>
<td>12</td>
<td>$65,000</td>
</tr>
<tr>
<td>Bob</td>
<td>010</td>
<td>1</td>
<td>$40,000</td>
</tr>
<tr>
<td>Charlie</td>
<td>011</td>
<td>20</td>
<td>$70,000</td>
</tr>
<tr>
<td>Debbie</td>
<td>100</td>
<td>30</td>
<td>$80,000</td>
</tr>
<tr>
<td>Evan</td>
<td>101</td>
<td>4</td>
<td>$50,000</td>
</tr>
<tr>
<td>Frank</td>
<td>110</td>
<td>8</td>
<td>$58,000</td>
</tr>
</tbody>
</table>

Request denied!
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>UID</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>001</td>
<td>12</td>
<td>$65,000</td>
</tr>
<tr>
<td>Bob</td>
<td>010</td>
<td>1</td>
<td>$40,000</td>
</tr>
<tr>
<td>Charlie</td>
<td>011</td>
<td>20</td>
<td>$70,000</td>
</tr>
<tr>
<td>Debbie</td>
<td>100</td>
<td>30</td>
<td>$80,000</td>
</tr>
<tr>
<td>Evan</td>
<td>101</td>
<td>4</td>
<td>$50,000</td>
</tr>
<tr>
<td>Frank</td>
<td>110</td>
<td>8</td>
<td>$58,000</td>
</tr>
</tbody>
</table>

Give me the list of all names

Alice
Bob
Charlie
Debbie
Evan
Frank

Give me all UIDs and salaries

(Alice, 001)
(Bob, 010)
(Charlie, 011)
(Debbie, 100)
(Evan, 101)
(Frank, 110)

(001, $65,000)
(010, $40,000)
(011, $70,000)
(100, $80,000)
(101, $50,000)
(110, $58,000)
Example

<table>
<thead>
<tr>
<th>Name</th>
<th>UID</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>001</td>
<td>12</td>
<td>$65,000</td>
</tr>
<tr>
<td>Bob</td>
<td>010</td>
<td>1</td>
<td>$40,000</td>
</tr>
<tr>
<td>Charlie</td>
<td>011</td>
<td>20</td>
<td>$70,000</td>
</tr>
<tr>
<td>Debbie</td>
<td>100</td>
<td>30</td>
<td>$80,000</td>
</tr>
<tr>
<td>Evan</td>
<td>101</td>
<td>4</td>
<td>$50,000</td>
</tr>
<tr>
<td>Frank</td>
<td>110</td>
<td>8</td>
<td>$58,000</td>
</tr>
</tbody>
</table>

External knowledge:
more years ⇒ higher pay

Some solutions

- In general, an unsolved problem
- Some techniques to mitigate the problem
  - Inference during database design
    - E.g., recognize dependencies between columns
    - Split data across several databases (next slide)
  - Inference detection at query time
    - Store the set of all queries asked by a particular user, and look for disallowed inferences before answering any query
    - Note: will not prevent collusion among multiple users
    - Can also store the set of all queries asked by anyone, and look for disallowed inference there
- As always, tradeoff security and usability
Using several databases

- DB1 stores (name, address), accessible to all
- DB2 stores (UID, salary), accessible to all
- DB3 stores (name, UID), accessible to admin

What if I want to add data for “start-date” (and make it accessible to all)?
- Adding to DB2 can be problematic (why?)
- Adding to DB1 seems ok (can we prove this?)

Statistical databases

- Database that only provides data of a statistical nature (average, standard deviation, etc.)
  - Pure statistical database: only stores statistical data
  - Statistical access to ordinary database: stores all data but only answers statistical queries
  - Focus on the second type

- Aim is to prevent inference about any particular piece of information
  - One might expect that by limiting to aggregate information, individual privacy can be preserved
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'

<table>
<thead>
<tr>
<th>Name</th>
<th>Gender</th>
<th>Years of service</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<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>

Alice's salary: $65,000

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’\Delta S| 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+\( \Delta \)

Query restriction

• 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

\[\begin{array}{ccc}
\text{deny} & \text{sum}(x, y, z) & \text{respond?} \\
\text{respond?} & \text{max}(x, y, z) & \text{deny?}
\end{array}\]
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
Belief tracking

- Keep track of attacker's knowledge, making assumptions about the attacker's initial knowledge
  - Revise after each query is answered

- Refuse to answer any queries that would raise user's knowledge above some threshold

- Again, need to be careful with refusals revealing information
  - Deny if there is any secret for which the answer would reveal information

Again: 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)
Perturbation

• Purposely add “noise”
  – Data perturbation: add noise to entire table, then answer queries accordingly (or release entire perturbed dataset)
  – Output perturbation: keep table intact, but add noise to answers

Perturbation

• Trade-off between privacy and utility!
  – No randomization – bad privacy but perfect utility
  – Complete randomization – perfect privacy but zero utility
Data perturbation

• One technique: data swapping
  – Substitute and/or swap values, while maintaining low-order statistics

<table>
<thead>
<tr>
<th>Gender</th>
<th>Major</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>Bio</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>3.0</td>
</tr>
<tr>
<td>F</td>
<td>Psych</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>Bio</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>CS</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td>EE</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>Psych</td>
<td>3.0</td>
</tr>
</tbody>
</table>

Restriction to any two columns is identical

Data perturbation

• Second technique: (re)generate the table based on derived distribution
  – For each sensitive attribute, determine a probability distribution that best matches the recorded data
  – Generate fresh data according to the determined distribution
  – Populate the table with this fresh data

• Queries on the database can never “learn” more than what was learned initially
Data perturbation

- Data cleaning/scrubbing: remove sensitive data, or data that can be used to breach anonymity
- $k$-anonymity: ensure that any “personally identifying information” is shared by at least $k$ members of the database
- Example...

Example: 2-anonymity

<table>
<thead>
<tr>
<th>Race</th>
<th>ZIP</th>
<th>Smoke?</th>
<th>Cancer?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asian</td>
<td>0213x</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Asian</td>
<td>0213x</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Asian</td>
<td>0214x</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Asian</td>
<td>0214x</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Black</td>
<td>0213x</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Black</td>
<td>0213x</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Black</td>
<td>0214x</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Black</td>
<td>0214x</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>White</td>
<td>0213x</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>White</td>
<td>0213x</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>White</td>
<td>0214x</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>White</td>
<td>0214x</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
Problems with k-anonymity

• Hard to find the right balance between what is “scrubbed” and utility of the data
• Not clear what security guarantees it provides
  – For example, what if I know that the Asian person in ZIP code 0214x smokes?
    • Does not deal with out-of-band information
  – What if all people who share some identifying information share the same sensitive attribute?

Output perturbation

• One approach: replace the query with a perturbed query, then return an exact answer to that
  – E.g., a query over some set of entries C is answered using some (randomly-determined) subset $C' \subseteq C$
  – User learns only the answer, not $C'$
• Second approach: add noise to the exact answer (to the original query)
  – E.g., answer $\text{SUM}(\text{salary}, S)$ with $\text{SUM}(\text{salary}, S) + \text{noise}$
A negative result [Dinur-Nissim]

• Heavily paraphrased:
  “Given a database with $n$ rows, if $O(n)$ queries are made to the database then essentially the entire database can be reconstructed even if $O(n^{1/2})$ noise is added to each answer”

• On the positive side, it is known that very small error can be used when the total number of queries is kept small

Formally defining privacy

• A problem inherent in all the approaches we have discussed so far (and the source of many of the problems we have seen) is that no definition of “privacy” is offered

• Recently, there has been work addressing exactly this point
  – Developing definitions
  – Provably secure schemes!
A definition of privacy

- *Differential privacy [Dwork et al.]*
- Roughly speaking:
  - For each row $r$ of the database (representing, say, an individual), the distribution of answers when $r$ is included in the database is “close” to the distribution of answers when $r$ is not included in the database
  - No reason for $r$ not to include themselves in the database!
  - Note: can’t hope for “closeness” better than $1/|DB|$ 
- Further refining/extending this definition, and determining when it can be applied, is an active area of research

Achieving privacy

- A “converse” to the Dinur-Nissim result is that adding *some* (carefully-generated) noise, and limiting the number of queries, *can* be proven to achieve privacy
Achieving privacy

• E.g., answer $\text{SUM}(\text{salary}, S)$ with $\text{SUM}(\text{salary}, S) + \text{noise}$, where the magnitude of the noise depends on the range of plausible salaries (but not on $|S|$!)
• Automatically handles multiple (arbitrary) queries, though privacy degrades as more queries are made
• Gives formal guarantees

Electronic Cash