Building Privacy-Aware Database Systems

CS848 Winter 2021
Module 1
This course will explore ...

- How to define a good privacy promise?
- How to design a privacy-preserving algorithms?
- How to build a privacy-aware database systems?

Greatly depend on the architecture setup and trust assumptions

- Client-server with trusted data curator
- Data federation
- Cloud service provider
Outline

• Part I: Differential Privacy Basics

• Part II: Implementation Challenges

• Upcoming Papers and Announcements
Desiderata for a Privacy Definition

1. Resilience to background knowledge
   – A privacy mechanism must be able to protect individuals’ privacy from attackers who may possess background knowledge

2. Privacy without obscurity
   – Attacker must be assumed to know the algorithm used as well as all parameters [MK15]

3. Post-processing
   – Post-processing the output of a privacy mechanism must not change the privacy guarantee [KL10, MK15]

4. Composition over multiple releases
   – Allow a graceful degradation of privacy with multiple invocations on the same data [DN03, GKS08]
Differential Privacy

• “An algorithm satisfies differential privacy (DP) if its output is insensitive to adding, removing or changing one record in its input database”

\[ M(D) \neq M(D') \]
Differential Privacy

[For every output ...]

For every pair of inputs that differ in one row

\[
\ln \left( \frac{\Pr[A(D_1) = o]}{\Pr[A(D_2) = o]} \right) \leq \varepsilon, \quad \varepsilon > 0
\]

For every output ...

Adversary should not be able to distinguish between any \( D_1 \) and \( D_2 \) based on any \( O \)

[Dwork ICALP 2006]
Why pairs of datasets *that differ in one row*?

For every pair of inputs that differ in one row

\[ D_1 \quad D_2 \]

Simulate the presence or absence of a single record

For every output ...

\[ O \]
Why *all* pairs of datasets ...?

For every pair of inputs that differ in one row

For every output ...

Guarantee holds no matter what the other records are.
Why *all* outputs?

\[ D_1 \]

\[ D_2 \]

\[ A(D_1) = O_1 \]

\( P \left[ A(D_1) = O_1 \right] \)

\[ \cdot \]

\[ \cdot \]

\[ \cdot \]

\[ A(D_2) = O_k \]

\( P \left[ A(D_2) = O_k \right] \)

Set of all outputs
Should not be able to distinguish whether input was $D_1$ or $D_2$ no matter what the output.
Privacy Parameter $\varepsilon$

For every pair of inputs that differ in one row

$D_1$  $D_2$  $O$

For every output …

$$\Pr[A(D_1) = o] \leq e^\varepsilon \Pr[A(D_2) = o]$$

Controls the degree to which $D_1$ and $D_2$ can be distinguished.
Smaller the $\varepsilon$ more the privacy (and worse the utility)
Laplace Mechanism

Aggregate Query: \( q \)

Noisy Answer

\[ \tilde{q}(D) = q(D) + \text{Lap} \left( \frac{GS(q)}{\varepsilon} \right) \]

Global Sensitivity

\[ \text{Lap}(\lambda): h(\eta) \propto \exp \left( -\frac{|\eta|}{\lambda} \right) \]

Private Database

e.g., COUNT

[DMNS 06]
How much noise for privacy?

Global Sensitivity of a query $q$ that outputs a real number: the maximum change to the query output, for any neighboring tables $D_1, D_2$ that differ in a row,

$$GS(q) = \max_{\forall \text{neighbor}(D_1, D_2)} |q(D_1) - q(D_2)|$$

$$= \max_{D_2 \in \text{dom}} \max_{\forall D_1 \in \text{neighbors}(D_2)} |q(D_1) - q(D_2)|$$

Any possible database $D_2$  
Add/remove any record from $D_2$

Theorem: $q(D) + \text{Lap}\left(\frac{GS(q)}{\varepsilon}\right)$ satisfies $\varepsilon$-DP.
Global Sensitivity: COUNT query

• # of people having flu?
• Sensitivity = 1

Solution: $2 + \eta$
where $\eta$ is drawn from $\text{Lap}(\frac{1}{\epsilon})$
  – Mean = 0
  – Variance = $2/\epsilon^2$
Global Sensitivity: SUM query

• Total usage of drug X?

• Suppose all values x are in [a,b]

• Sensitivity = b

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Age</th>
<th>Disease</th>
<th>Drug X</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>6'2&quot;</td>
<td>56</td>
<td>Cancer</td>
<td>3.5</td>
</tr>
<tr>
<td>F</td>
<td>5'3&quot;</td>
<td>30</td>
<td>Diabetes</td>
<td>2.3</td>
</tr>
<tr>
<td>F</td>
<td>5'9&quot;</td>
<td>24</td>
<td>Healthy</td>
<td>1.0</td>
</tr>
<tr>
<td>M</td>
<td>5'3&quot;</td>
<td>36</td>
<td>Flu</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>6'7&quot;</td>
<td>22</td>
<td>Flu</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Privacy of Laplace Mechanism

• Consider neighboring databases \( D \) and \( D' \)
• Consider some output \( O \)

\[
\frac{\Pr[A(D) = O]}{\Pr[A(D') = O]} = \frac{\Pr[q(D) + \eta = O]}{\Pr[q(D') + \eta' = O]}
= \frac{\Pr[\eta = O - q(D)]}{\Pr[\eta' = O - q(D')]}
= \frac{e^{-|O-q(D)|/\lambda}}{e^{-|O-q(D')|/\lambda}}
\leq e^{\frac{|q(D) - q(D')|}{\lambda}} \leq e^{\frac{GS(q)}{\lambda}} = e^\varepsilon
\]
Utility of Laplace Mechanism

• Laplace mechanism works for any function that returns a real number

• Error: $E(\text{true answer} - \text{noisy answer})^2$

$$= \text{Var}(\text{Lap}(GS(q)/\varepsilon))$$

$$= \frac{2 \times GS(q)^2}{\varepsilon^2}$$
Sequential Composition

• If $M_1, M_2, \ldots, M_k$ are algorithms that access a private database $D$ such that each $M_i$ satisfies $\varepsilon_i$-differential privacy,

then the combination of their outputs satisfies $\varepsilon$-differential privacy with

$$\varepsilon = \varepsilon_1 + \ldots + \varepsilon_k$$
Postprocessing

• If $M$ is an $\varepsilon$-differentially private algorithm, any additional post-processing $A \circ M$ also satisfies $\varepsilon$-differential privacy.
Outline

• Part I: Differential Privacy Basics
  – Laplace mechanism
  – Global sensitivity analysis

• Part II: Implementation Challenges

• Upcoming Papers and Announcements
Global Sensitivity: Vector of Counts

• \( q \): # of males and # of females with flu?
  – Return \( \left( \frac{c_m}{c_f} \right) + \left( \frac{\eta_1}{\eta_2} \right) \), where \( \eta_i \sim Lap \left( \frac{GS(q)}{\epsilon} \right) \)

\[
GS(q) = \max_{\forall \text{ neighbor}(D_1,D_2)} \|q(D_1) - q(D_2)\|_1
\]

• Sensitivity = ?
  \[
  \max \left( \|(+1)\|_1, \|-1\|_1, \|(+0)\|_1, \|-0\|_1 \right)
  
\]

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Age</th>
<th>Disease</th>
<th>Drug X</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>6'2&quot;</td>
<td>56</td>
<td>Cancer</td>
<td>3.5</td>
</tr>
<tr>
<td>F</td>
<td>5'3&quot;</td>
<td>30</td>
<td>Diabetes</td>
<td>2.3</td>
</tr>
<tr>
<td>F</td>
<td>5'9&quot;</td>
<td>24</td>
<td>Healthy</td>
<td>1.0</td>
</tr>
<tr>
<td>M</td>
<td>5'3&quot;</td>
<td>36</td>
<td>Flu</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>6'7&quot;</td>
<td>22</td>
<td>Flu</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Global Sensitivity: Complex Queries

• # of (distinct) diseases?
• # of rare diseases (appear < 5 times) and # of common diseases (appear >= 5 times)?
• ...

How to automate this process for a large class of queries?

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Age</th>
<th>Disease</th>
<th>Drug X</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>6'2&quot;</td>
<td>56</td>
<td>Cancer</td>
<td>3.5</td>
</tr>
<tr>
<td>5'3&quot;</td>
<td>30</td>
<td></td>
<td>Diabetes</td>
<td>2.3</td>
</tr>
<tr>
<td>5'9&quot;</td>
<td>24</td>
<td></td>
<td>Healthy</td>
<td>1.0</td>
</tr>
<tr>
<td>5'3&quot;</td>
<td>36</td>
<td></td>
<td>Flu</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>6'7&quot;</td>
<td>22</td>
<td>Flu</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Transformations & Stability

• Express a counting query as a sequence of pre-defined transformations
  – \( q(D): V_m \ldots ((V_2(V_1(D)))) \)

• Track the stability of each transformation \( s_V \):
  – Maximum number of rows in \( V \) that can change due to changing a single row in the input table

• Bound global sensitivity \( GS(q) \leq s_{V_m} \ldots s_{V_2} s_{V_1} \)
Transformations & Stability

- # of (distinct) diseases?

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Age</th>
<th>Disease</th>
<th>Drug X</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>6'2&quot;</td>
<td>56</td>
<td>Cancer</td>
<td>3.5</td>
</tr>
<tr>
<td>F</td>
<td>5'3&quot;</td>
<td>30</td>
<td>Cancer</td>
<td>2.3</td>
</tr>
<tr>
<td>F</td>
<td>5'9&quot;</td>
<td>24</td>
<td>Healthy</td>
<td>1.0</td>
</tr>
<tr>
<td>M</td>
<td>5'3&quot;</td>
<td>36</td>
<td>Flu</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>6'7&quot;</td>
<td>22</td>
<td>Flu</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Transformations & Stability

• # of (distinct) diseases?

<table>
<thead>
<tr>
<th>Sex</th>
<th>Height</th>
<th>Age</th>
<th>Disease</th>
<th>Drug X</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>6'2&quot;</td>
<td>56</td>
<td>Cancer</td>
<td>3.5</td>
</tr>
<tr>
<td>F</td>
<td>5'3&quot;</td>
<td>30</td>
<td>Cancer</td>
<td>2.3</td>
</tr>
<tr>
<td>F</td>
<td>5'9&quot;</td>
<td>24</td>
<td>Healthy</td>
<td>1.0</td>
</tr>
<tr>
<td>M</td>
<td>5'3&quot;</td>
<td>36</td>
<td>Flu</td>
<td>4.0</td>
</tr>
<tr>
<td>M</td>
<td>6'7&quot;</td>
<td>22</td>
<td>Flu</td>
<td>2.2</td>
</tr>
</tbody>
</table>
Transformations & Stability

• Frequency of the word “covid”

<table>
<thead>
<tr>
<th>Tweet id</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id 1</td>
<td>“privacy”</td>
</tr>
<tr>
<td>Id 1</td>
<td>“covid”</td>
</tr>
<tr>
<td>Id 1</td>
<td>“covid”</td>
</tr>
<tr>
<td>Id 2</td>
<td>“mask”</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Id 5</td>
<td>“risk”</td>
</tr>
<tr>
<td>Id 5</td>
<td>“covid”</td>
</tr>
</tbody>
</table>

\[ s_{V_2} = 1 \]

Remove irrelevant rows and columns

\[ s_{V_1} = ? \]

Split rows

<table>
<thead>
<tr>
<th>Tweet id</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id 1</td>
<td>{“privacy”, “covid”,...}</td>
</tr>
<tr>
<td>Id 2</td>
<td>{“mask”, “covid”,...}</td>
</tr>
<tr>
<td>Id 3</td>
<td>{“covid”, “covid”,...}</td>
</tr>
<tr>
<td>Id 4</td>
<td>{“privacy”, “risk”,...}</td>
</tr>
<tr>
<td>Id 5</td>
<td>{“water”, “plant”,...}</td>
</tr>
</tbody>
</table>

GS: unbounded

\[ \text{Count}(V_2) + \text{Lap}(\frac{GS}{\epsilon}) \]

\[ V_1 \rightarrow V_2 \]
Transformations & Stability

• Frequency of the word “covid”

<table>
<thead>
<tr>
<th>Tweet id</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id 1</td>
<td>“privacy”</td>
</tr>
<tr>
<td>Id 1</td>
<td>“covid”</td>
</tr>
<tr>
<td>Id 1</td>
<td>…</td>
</tr>
<tr>
<td>Id 2</td>
<td>“mask”</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Id 5</td>
<td>“risk”</td>
</tr>
<tr>
<td>Id 5</td>
<td>…</td>
</tr>
</tbody>
</table>

$V_1$

<table>
<thead>
<tr>
<th>Tweet id</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id 1</td>
<td>“covid”</td>
</tr>
<tr>
<td>Id 2</td>
<td>“mask”</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
<tr>
<td>Id 5</td>
<td>“risk”</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

$V_2$

$GS: 1 \cdot k = k$

Count($V_2$) + Lap($\frac{GS}{\epsilon}$)

$V_2$

<table>
<thead>
<tr>
<th>Tweet id</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id 1</td>
<td>{“privacy”, “covid”, …}</td>
</tr>
<tr>
<td>Id 2</td>
<td>{“mask”, “covid”, …}</td>
</tr>
<tr>
<td>Id 3</td>
<td>{“covid”, “covid”, …}</td>
</tr>
<tr>
<td>Id 4</td>
<td>{“privacy”, “risk” …}</td>
</tr>
<tr>
<td>Id 5</td>
<td>{“water”, “plant” …}</td>
</tr>
</tbody>
</table>

$D$

$s_{V_2} = 1$

Remove irrelevant rows and columns

$s_{V_1} = ?$

Take first $k$ words per row

$s_{V_1} = k$

Split rows

$\mathbf{GS}: 1 \cdot k = k$
Outline

• Part I: Differential Privacy Basics
  – Laplace mechanism
  – Global sensitivity analysis
  – PINQ: Implementation of DP

• Part II: Implementation Challenges

• Upcoming Papers and Announcements
PINQ (Privacy Integrated Queries)

- Implementation is based on C#'s LINQ language

**Example 1** Counting searches from distinct users in PINQ.

```csharp
var data = new PINQueryable<SearchRecord>(... ...);

var users = from record in data
             where record.Query == argv[0]
             groupby record.IPAddress

Console.WriteLine(argv[0] + ": " + users.NoisyCount(0.1));
```

**Example 3** [Abbreviated] Implementation of NoisyCount.

```csharp
double NoisyCount(double epsilon) {
    if (myagent.Alert(epsilon))
        return mysource.Count() + Laplace(1.0/epsilon);
    else
        throw new Exception("Access is denied");
}
```

Transformations

Track privacy budget
PINQAgent

- Keeps track of privacy budget

**Example 2** Implementing a fixed budget in a PINQAgent.

```java
public class PINQAgentBudget : PINQAgent
{
    private double budget;

    public override bool Alert(double epsilon)
    {
        if (budget < epsilon)
            return false;

        budget = budget - epsilon;
        return true;
    }

    public PINQAgentBudget(double b) { budget = b; }
}
```
PINQ: Aggregation Operators

• Laplace Mechanism
  – NoisyCount
  – NoisySum

• Exponential Mechanism
  – NoisyMedian
  – NoisyAverage
PINQ: Transformation Operators

- Aggregations are computed on transformed data
  - *Where*: takes as input a predicate (arbitrary C# function), and outputs a subset of the data satisfying the predicate  
    - Stability = 1

- *Select*: Maps each input record into a different record using a C# function  
  - Stability = 1

- *GroupBy*: Groups records by key values  
  - Stability = ?

- *Join*: Takes two datasets, and key values for each and returns groups of pairs of records for each key.
Join Operator

- `SELECT COUNT(*) FROM A JOIN B ON A.k = B.k`
Consider All Possible DB Pairs

- **SELECT COUNT(*) FROM A JOIN B ON A.k = B.k**

Unbounded change in join output size
Special “Join” in PINQ

- SELECT COUNT(*) FROM A JOIN B ON A.k = B.k

Keep one copy for each key value

Limited change to join output size

Limited queries or poor utility !!!
Outline

• Part I: Differential Privacy Basics
  – Laplace mechanism
  – Global sensitivity analysis
  – PINQ: Implementation of DP

• Part II: Implementation Challenges
  – Highly sensitive queries (Joins)
  – Attacks on DP Implementations

• Upcoming Papers and Announcements
Local Sensitivity of True Database

- **SELECT COUNT(\(*) FROM A JOIN B ON A.k = B.k**

<table>
<thead>
<tr>
<th>k</th>
<th>v</th>
<th>k</th>
<th>v</th>
<th>k</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td></td>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td>1</td>
<td>a</td>
</tr>
</tbody>
</table>

Can we just add noise proportional to LS? **No**

A | B | A JOIN B
---|---|---
| k | v | k | v |
| 1 | a | 1 | a |
| 1 | b | 1 | b |

Add a row to Table A

+2 in join output size for this given instance

Local sensitivity = 2
Local Sensitivity v.s. Global Sensitivity

• **Local sensitivity:**
  – max change in query output when adding/removing a row from the true database instance $D$

  \[
  LS(q,D) = \max_{\forall D_1 \in \text{neighbors}(D)} |q(D) - q(D_1)|
  \]

• **Global sensitivity:**
  – independent of the true database instance

  \[
  GS(q) = \max_{D_2 \in \text{dom}} \max_{\forall D_1 \in \text{neighbors}(D_2)} |q(D_1) - q(D_2)|
  \]
Local Sensitivity v.s. Global Sensitivity

• Local sensitivity:
  – max change in query output when adding/removing a row from the true database instance $D$

$$LS(q,D) = \max_{\forall D_1 \in \text{neighbors}(D)} |q(D) - q(D_1)|$$

• Global sensitivity:
  – independent of the true database instance

$$GS(q) = \max_{D_i \in \text{dom}} LS(q, D_i)$$
Smooth Sensitivity (SS)

\[ SS(q,D) = \max_{D_i \in \text{dom}} e^{-\beta \cdot d(D_i,D)} LS(q,D_i) \]

Decay with the distance from true database

Theorem: \( q(D) + \text{Lap}\left(\frac{2SS(q,D)}{\epsilon}\right) \) satisfies \((\epsilon, \delta)\)-DP, where \( \beta = \frac{\epsilon}{2 \ln \frac{2}{\delta}} \)

[Nissim et al., STOC 2007]
Elastic Sensitivity (ES)

- Computing LS and SS is computationally expensive

- Elastic sensitivity: loose upper bound of local sensitivity instead of exact local sensitivity  [FLEX]

\[ ES(q,D) = \max_{D_i \in \text{dom}} e^{-\beta \cdot d(D_i,D)} \cdot LS'(q,D_i) \]

- Recursively compute upper bound of \(LS(q,D_i)\) for \(D_i\) at distance \(d\) from the true database

\[ ES(q,D) = \max_{d=0,1,...} e^{-\beta \cdot d} \max_{\text{dist}(D,D_i)=d} LS'(q,D_i) \]
Elastic Sensitivity (ES)

- Express each query as query plan (relational algebra)
- Recursive computation of statistics
  - max frequency and elastic stability

Figure 1: (a) syntax of core relational algebra; (b) definition of elastic stability and elastic sensitivity at distance $k$; (c) definition of maximum frequency at distance $k$; (d) definition of ancestors of a relation.
Comparisons

At worst case, the local sensitivity of the true database instance is as large as the global sensitivity, all these approaches need to add a large amount of noise!
Other Approaches

• Apply a transformation $V()$ over the database instance $D$, such that
  – the sensitivity of $q(V(D))$ is small, and
  – $q(V(D))$ is close to $q(D)$
  – E.g. transformation example on slide 23, slide 35

• Prior work
  – Sample & Aggregate (GUPT, Mohan et al. SIGMOD’12)
  – Truncation (PrivateSQL, Kotsogiannis et al. VLDB 2019)
Outline

• Part I: Differential Privacy Basics
  – Laplace mechanism
  – Global sensitivity analysis
  – PINQ: Implementation of DP

• Part II: Implementation Challenges
  – Highly sensitive queries (Joins)
  – Attacks on DP Implementations

• Upcoming Papers and Announcements
Covert Channel

• Key assumption in DP implementations: The querier can only observe the result of the query, and nothing else.
  – This answer is guaranteed to be differentially private.

• In practice: The querier can observe other effects.
  – E.g. Time taken by the query to complete, power consumption, etc.
  – Suppose a system takes 1 minute to answer a query if Bob has cancer and 1 microsecond otherwise, then based on query time the adversary may know that Bob has cancer.
Threat Model

• Assume the adversary (querier) does not have physical access to the machine.
  – Poses queries over a network connection.

• Given a query, the adversary can observe:
  – Answer to their question
  – Time that the response arrives at their end of the connection
  – The system’s decision to execute the query or deny (since the new query would exceed the privacy budget)
Timing Attack

Function is_f(Record r)
{
    if(r.name = Bob && r.disease = Cancer)
        sleep(10 sec);  // or go into infinite loop, or throw exception
    return f(r);
}

Function countf()
{
    var fs = from record in data
        where (is_f(record))
    print fs.NoisyCount(0.1);
}

If Bob has Cancer, then the query takes > 10 seconds; otherwise, the query takes less than a second
Global Variable Attack

Boolean found = false;
Function f(Record r) {
    if (found) return 1;
    if (r.name = Bob && r.disease = Cancer) {
        found = true; return 1;
    } else return 0;
}

Function countf() {
    var fs = from record in data
        where (f(record))
    print fs.NoisyCount(0.1);
}
Privacy Budget Attack

Function is_f(Record r){
    if(r.name == Bob && r.disease == Cancer){
        run a sub-query that uses a lot of the privacy budget;
    }
    return f(r);
}

Function countf(){
    var fs = from record in data where (f(record))
    print fs.NoisyCount(0.1);
}

If Bob does not have Cancer, then privacy budget decreases by 0.1.
If Bob has Cancer, then privacy budget decreases by 0.1 + Δ.
Even if adversary can’t query for the budget, he can detect the change in budget by counting how many more queries are allowed.
Avoid Covert Channel Attacks

- **Fuzz:**
  - Global variables are not supported in this language, thus ruling out our *state attacks*.
  - **Type checker** rules out *budget-based channels* by statically checking the sensitivity of a query *before* they are executed.
  - **Predictable query processor** ensures that each microquery takes the same amount of time, ruling out *timing attacks*. 

Handling Timing Attacks

• Each microquery takes exactly the same time $T$
  – If it takes less time – delay the query
  – If it takes more time – abort the query

• But this can leak information!
  – Wrong Solution

• If it takes more time – return a default value
Summary

• Part I: Differential Privacy Basics
  – Laplace mechanism
  – Global sensitivity analysis
    • Transformations and stability (first proposed in PINQ)
  – PINQ: Implementation of DP

• Part II: Implementation Challenges
  – Highly sensitive queries (Joins)
    • Global sensitivity, local sensitivity, smooth sensitivity, elastic sensitivity (FLEX)
  – Attacks on DP Implementations
    • Timing, global variable, privacy budget (Fuzz)
Paper Readings

• Week 3
  – 1a. PrivateSQL
  – 1b. Airavat
  – 1c. VideoDP

• Week 4
  – 1d. DP for Streams
  – 1e. DP for Growing DB
  – 1f. Formalize Data Deletion
Announcement

• Paper reviews (next week)
  – Submit on Learn by Tue 9am

• Paper presentation
  – Either during the Tue live session
  – Or upload a video to Learn by Mon, 11pm

• Assignment 1
  – Release after Wed session
  – Latex file will be available
  – Submit pdf on Learn, by Feb 1, 11pm
Paper Review Guideline

• ½ page – 1 page
• A summary of the paper:
  – Motivation, problem, approach, result
• 3 Strengths
• 3 Weaknesses
Paper Presentation Guideline

• A summary of the paper
  • Motivation, problem, approach, result

• One technical piece in the paper
  • e.g. how result was reached

• One slide on points for discussion/open questions
Discussion Time