Lecturers
Prof. Dr. Florina Ciorba. (mailto:florina.ciorba@unibas.ch) Office 06.007
Prof. Dr. Heiko Schuldt (mailto:heiko.schuldt@unibas.ch) Office 06.005
Prof. Dr. Christian Tschudin (mailto:christian.tschudin@unibas.ch) Office 06.002
Prof. Dr. Isabel Wagner (mailto:isabel.wagner@unibas.ch) Office 05.009

Assistant
Shiva Parsarad (mailto:shiva.parsarad@stud.unibas.ch)

Release: 11 November 2024
Deadline: 25 November 2024, 23:55 (ADAM)

Requirements

In this exercise, you will work with **differential privacy**, one of the most important state-of-the-art privacy mechanisms. You will implement several differentially private mechanisms and analyze their properties.
Requirements common to all tasks:

(a) Your implementations must be in Python 3.
(b) The only import statements you may use are:

```
import pandas as pd
import numpy as np
from scipy import stats
```

(c) Create a brief manual that explains how to run your code and interpret the output. If you do not provide this manual, you will not get any points.

The solutions for this exercise must be delivered in a zip or tar file containing:

- All source codes containing the solutions.
- A single PDF file containing the written answers and the manual.

1 Laplace mechanism and counting queries (4 points)

Write code to:

- Implement the Laplace mechanism in a function `laplace_mech(v, sensitivity, epsilon)`
- Implement a differentially private query that retrieves the number of people in the *adult* dataset who are over 29 (use $\epsilon = \ln 2$)

Explain:

- What is the sensitivity of this query, and why?

2 Contingency tables (4 points)

Write code to:

- Generate differentially private contingency tables for any dataset and any combination of two of its columns.
- Generate a differentially private contingency table with a total privacy cost of $\epsilon = 0.3$ for the *Relationship* and *Race* columns of the *adult* dataset.

Explain:

- Does parallel composition apply for generation of the contingency table? Why or why not?
- Does the number of variables used in constructing the contingency table matter for privacy cost? Does it matter for accuracy?

3 Differentially private selections from sets (4 points)

Write code to:

- Implement the function `score` that returns high scores for common occupations, and low scores for uncommon ones (e.g. the score could be the number of people with that occupation).
- Implement the function `most_common_occupation` that returns the most common occupation in a differentially private way using the Laplace mechanism.
- Compute the most common occupation for $\epsilon = 0.05$.

Explain:

- What is the sensitivity of your scoring function?
- What is the total privacy cost for `most_common_occupation` and why?

4 Differentially private sums (4 points)

Write code to:

- Implement the function `dp_sum_capgain`. The function should compute a differentially private sum of the *Capital Gain* column of the *adult* dataset, and have a total privacy cost of *epsilon*.
- Compute the differentially private sum of the *Capital Gain* column for $\epsilon = 0.04$.

Explain:

- What clipping parameter did you use in your definition of `dp_sum_capgain`, and why?
- What is the sensitivity of the query you used in `dp_sum_capgain`, and how is it bounded?
- Argue that your definition of `dp_sum_capgain` has a total privacy cost of epsilon.

5 Sensitivity (4 points)

Consider the following definition of a differencing attack (without differential privacy).

```
def differencing_attack():
        q1 = adult['Age'].sum()
        q2 = adult[adult['Name'] != 'Karrie Trusslove']['Age'].sum()
        return q1 - q2
print('Differencing attack result:', differencing_attack())
```

Explain:

- What is the sensitivity of the `differencing_attack` query defined above, and why?

Write code to:

- Implement the function `dp_differencing_attack` that uses the correct sensitivity and thus correctly satisfies differential privacy.