Data protection procedures

Vicenç Torra

November, 2022

Umeå University, Sweden

Outline

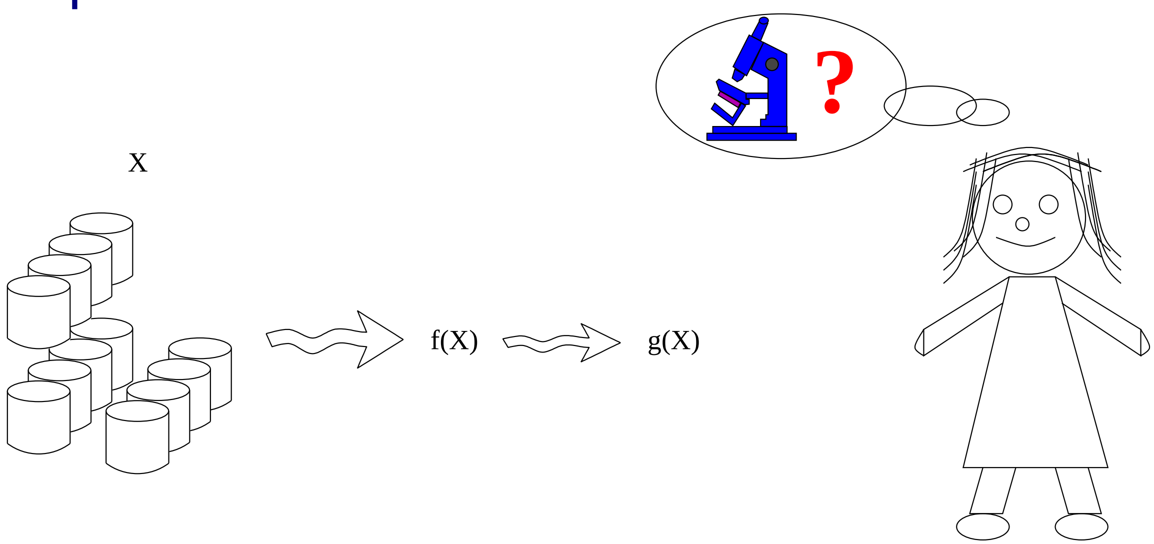
1. Computation-driven approaches
   - Differential privacy
   - Centralized approach: trusted third party
   - Distributed approach: secure multiparty computation

Computation-driven approaches

Computation-driven: Privacy model perspective

Privacy models. Computational definition. Computing a function

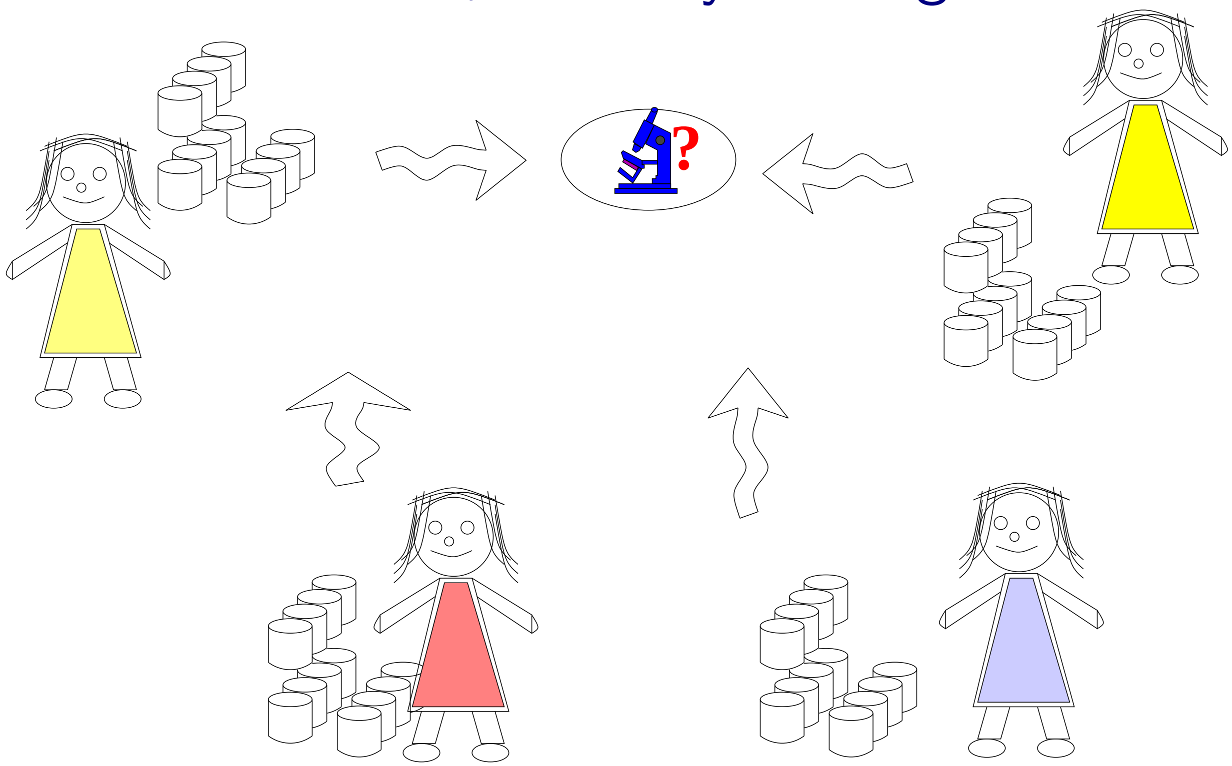
- **Differential privacy.** The output of a query to a database should not depend (much) on whether a record is in the database or not.
- **Integral privacy.** Inference on the databases. E.g., changes have been applied to a database.
- **Homomorphic encryption.** We want to avoid access to raw data and partial computations.



Computation-driven: Privacy model perspective

Privacy models. Computational definition. Computing a function

- **Secure multiparty computation.** Several parties want to compute a function of their databases, but only sharing the result.



Computation-driven: "Whose privacy" perspective

Respondent and owner privacy

- Data-driven or general-purpose
- Computation-driven or specific-purpose (Ch. 5)
  - Single database: differential privacy (Ch. 5.1)
  - Multiple databases:
    - Centralized approach: trusted third party
    - Distributed approach: secure multiparty computation (Ch. 5.2)
- Result-driven

Differential privacy

Differential privacy

- Computation-driven/single database
  - Privacy model: differential privacy[1]
  - We know the function/query to apply to the database: f

- Example:

  compute the mean of the attribute salary of the database for all those living in Town.

[1] There are other models as e.g. query auditing (determining if answering a query can lead to a privacy breach), and integral privacy

Differential privacy

- Differential privacy (Dwork, 2006).
  - Motivation:
    - ▷ the result of a query should not depend on the presence (or absence) of a particular individual
    - ▷ the impact of any individual in the output of the query is limited

  differential privacy ensures that the removal or addition of a single database item does not (substantially) affect the outcome of any analysis (Dwork, 2006)

Differential privacy

- Mathematical definition of differential privacy (in terms of a probability distribution on the range of the function/query)
  - A function K_q for a query q gives ϵ-differential privacy if for all data sets D_1 and D_2 differing in at most one element, and all $S \subseteq Range(K_q)$,

$$\frac{Pr[K_q(D_1) \in S]}{Pr[K_q(D_2) \in S]} \leq e^{\epsilon}.$$

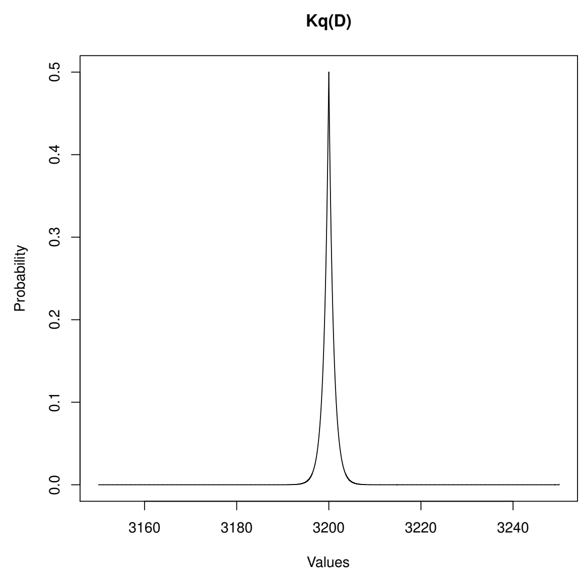
(with 0/0=1) or, equivalently,

$$Pr[K_q(D_1) \in S] \leq e^{\epsilon} Pr[K_q(D_2) \in S].$$

- ϵ is the level of privacy required (*privacy budget*). The smaller the ϵ, the greater the privacy we have.

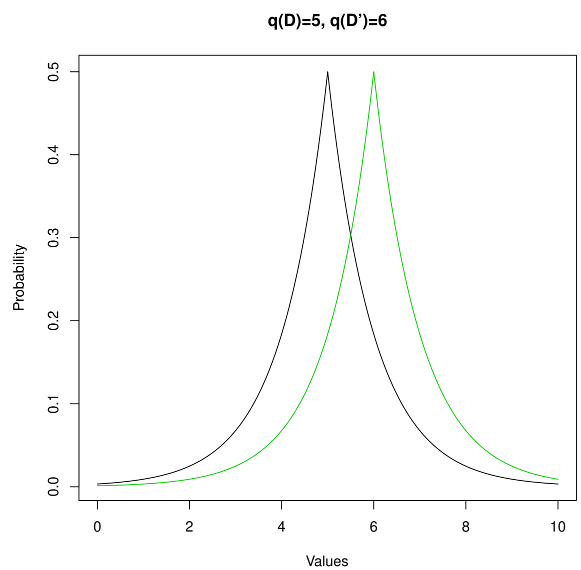
Differential privacy

- Differential privacy
  - A function K_q for a query q gives ϵ-differential privacy if . . .
    - ▷ $K_q(D)$ ~~is a constant. E.g.,~~
$$\cancel{K_q(D) = 0}$$
    - ▷ $K_q(D)$ is a randomized version of $q(D)$:
$$K_q(D) = q(D) + \textit{and some appropriate noise}$$



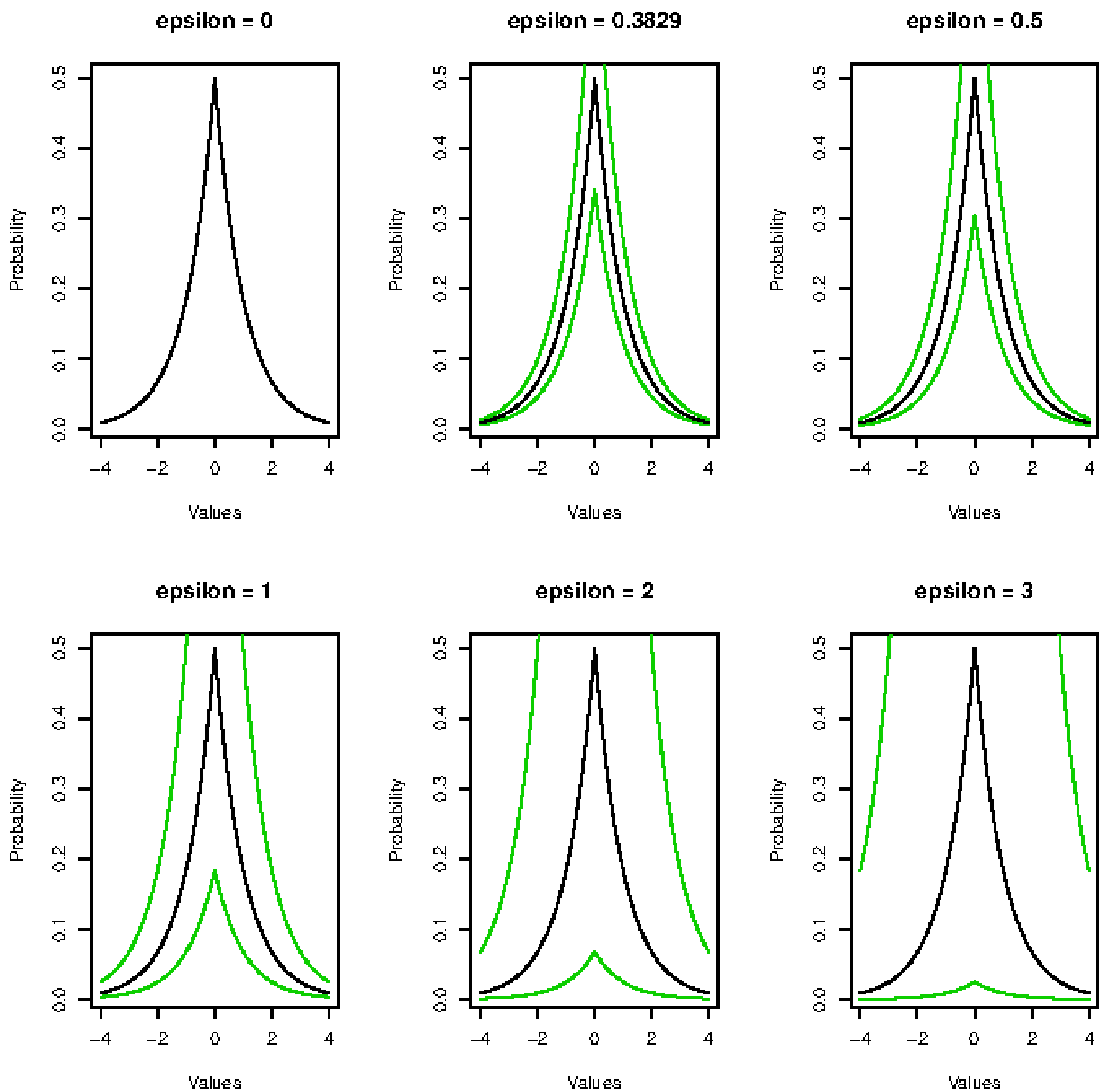
Differential privacy

- Differential privacy
  - $K_q(D)$ for a query q is a randomized version of $q(D)$
    - ▷ Given two neighbouring databases D and D' $K_q(D)$ and $K_q(D')$ should be similar enough . . .
  - Example with $q(D) = 5$ and $q(D') = 6$ and adding a Laplacian noise $L(0, 1)$



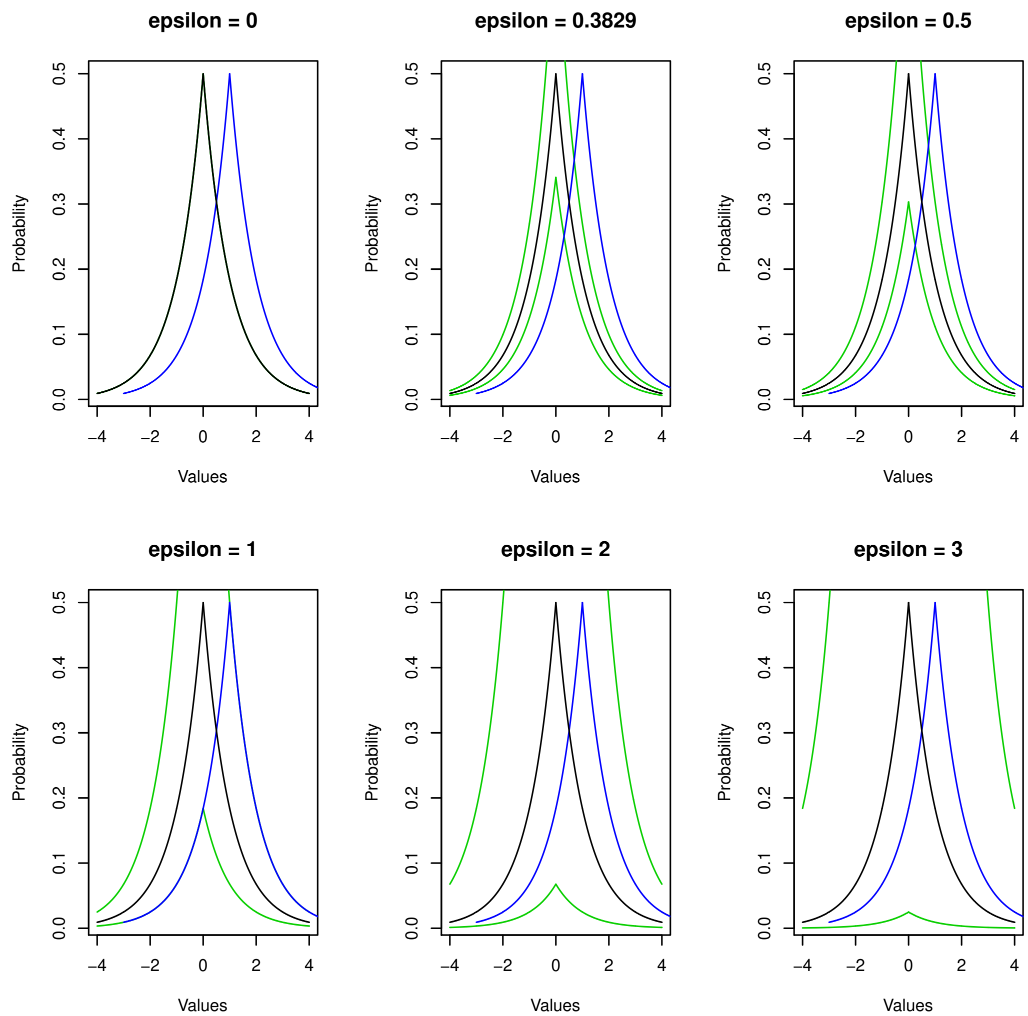
  - Let us compare different ϵ for noise following $L(0, 1)$ . . .

Differential privacy: comparing ϵ for $L(0,1)$



Differential privacy: Accepting 0+2? (using ϵ,$L(0,1)$)

Can $0+2$ be acceptable ? I.e., with a distribution similar enough?



Differential privacy

- These examples use the Laplace distribution $L(\mu, b)$.
  - I.e., probability density function:

$$f(x|\mu, b) = \frac{1}{2b} exp\left(-\frac{|x-\mu|}{b}\right)$$

    where
    - μ: location parameter
    - b: scale parameter (with $b > 0$)

- Properties
  - When $b = 1$, the function for $x > 0$ corresponds to the exponential distribution scaled by $1/2$.
  - Laplace has fatter tails than the normal distribution
  - When $\mu = 0$, for all translations $z \in \mathbb{R}$, $h(x+z)/h(x) \leq exp(|z|)$.

Differential privacy

- Implementation of differential privacy for a numerical query.
  - $K_q(D)$ is a randomized version of $q(D)$:
  $$K_q(D) = q(D) + \textit{and some appropriate noise}$$
  - What is *and some appropriate noise*

- Sensitivity of a query
  - Let $\mathcal{D}$ denote the space of all databases; let $q : \mathcal{D} \to \mathbb{R}^d$ be a query; then, the sensitivity of q is defined

$$\Delta_{\mathcal{D}}(q) = \max_{D, D' \in \mathcal{D}} ||q(D) - q(D')||_1.$$

where $|| \cdot ||_1$ is the L_1 norm, that is, $||(a_1, \ldots, a_d)||_1 = \sum_{i=1}^{d} |a_i|$.

- Definition essentially meaningful when data has upper & lower bounds

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - Sensitivity of the mean:

$$\Delta_{\mathcal{D}}(mean) = (max - min)/S$$

    where $[min, max]$ is the range of the attribute, and S is the minimal cardinality of the set.
    - ▷ If no assumption is made on the size of S: $\Delta_{\mathcal{D}}(mean) = (max - min)$
  - Parameter ϵ:

    (Lee, Clifton, 2011) recommend $\epsilon = 0.3829$ for the mean

Differential privacy

- Implementation of differential privacy for a numerical query.
  - Differential privacy via noise addition to the true response
  - Noise following a Laplace distribution $L(0, b)$ with mean equal to zero and scale parameter $b = \Delta(q)/\epsilon$.
    ($\Delta(q)$ is the sensitivity of the query)
  - **Algorithm** Differential privacy:
    - ▷ **Input:** D: Database; q: query; ϵ: parameter of differential privacy;
    - ▷ **Output:** Answer to the query q satisfying ϵ-differential privacy
    - ▷ $a := q(D)$ with the original data
    - ▷ $\Delta_{\mathcal{D}}(q)$:= the sensitivity of the query for a space of databases D
    - ▷ Generate a random noise r from a $L(0, b)$ where $b = \Delta(q)/\epsilon$
    - ▷ Return $a + r$

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - Example[2]:
    - ▷ $D = \{1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000\}$
      $\Rightarrow$ mean = 3300
    - ▷ Adding Ms. Rich's salary 100,000 Eur/month: mean = 12090,90 !
      (a extremely high salary changes the mean significantly)
      $\Rightarrow$ We infer Ms. Rich from Town was attending the unit

$\Rightarrow$ Differential privacy to solve this problem

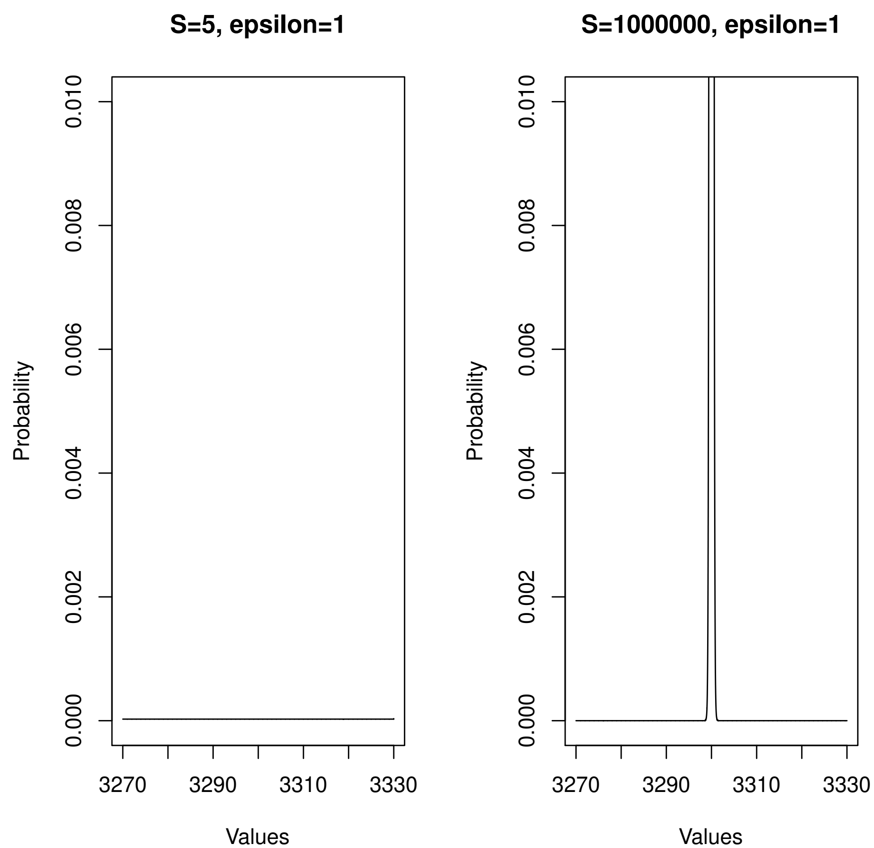
[2] Average wage in Ireland (2018): 38878 $\Rightarrow$ monthly 3239 Eur
https://www.frsrecruitment.com/blog/market-insights/average-wage-in-ireland/

Differential privacy

- Implementation of differential privacy: The case of the mean
  - Consider the mean salary
  - Range of salaries $[1000, 100000]$
- Compute for $\epsilon = 1$, assume that at least $S = 5$ records
  - sensitivity $\Delta_{\mathcal{D}}(q) = (max - min)/S = 19800$
  - scale parameter $b = 19800/1 = 19800$
  - For the database: (mean = 3300)
    D={1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000}
  - Output: $K_{mean}(D) = 3300 + L(0, 19800)$
- Compute for $\epsilon = 1$, assume that at least $S = 10^6$ records
  - sensitivity $\Delta_{\mathcal{D}}(q) = (max - min)/S = 0.099$
  - scale parameter $b = 0.099/1 = 0.099$
  - For the database: (mean = 3300)
    D={1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000}
  - Output: $K_{mean}(D) = 3300 + L(0, 0.099)$

Differential privacy: The two distributions

- Comparing
  - (i) $(S = 5, \epsilon = 1)$ $K_{mean}(D) = 3300 + L(0, 19800)$ and
  - (ii) $(S = 10^6, \epsilon = 1)$ $K_{mean}(D) = 3300 + L(0, 0.099)$



Differential privacy

- Laplace mechanism for differential privacy (numerical query)

$$K_q(D) = q(D) + L(0, \Delta(q)/\epsilon)$$

  - **Proposition.** For any function q, the Laplace mechanism satisfies ϵ-differential privacy.

Differential privacy

- Implementation of differential privacy: The case of the mean.
  - "Clamping down" on the output: (McSherry, 2009; Li, Lyu, Su, Yang, 2016 Sections 2.5.3 and 2.5.4)
    - ▷ The output of a query is within a range $[mn, mx]$ even if data is not. E.g., compute $q(D) = q'_{mn,mx}(mean(D))$ with q' as follows

$$q'_{mn,mx}(x) = \begin{cases} mn & \text{if } x < mn \\ x & \text{if } mn \leq x \leq mx \\ mx & \text{if } mx < x \end{cases}$$

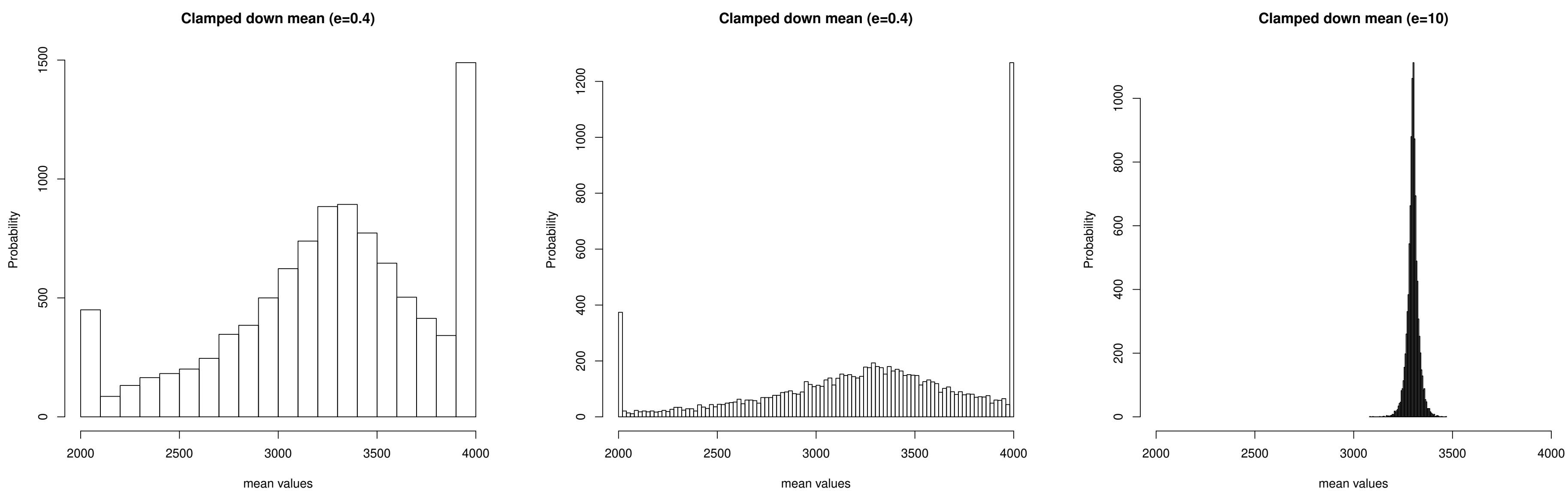
$\Rightarrow$ we can define ϵ-differential privacy for this query $q(D)$

Differential privacy

- Implementation of clamping-down mean
  - Differential privacy via noise addition to the true response
  - Arbitrary size S of the database D (i.e, $S = |D|$)
  - Output in the interval $[mn, mx]$
  - Solution and proof in (Li, Lyu, Su, Yang, 2016 Section 2.5.4)
  - **Algorithm** Differentially private clamping-down mean
    - **Input:** D: (one-dimensional) Database; $S : size$; ϵ: parameter of differential privacy; mn,mx: real
    - **Output:** A ϵ-differentially private mean
    - if $S = 0$ then
      $r :=$ uniform random in $[0, 1]$
      if $r < 1/2exp(-\epsilon/2)$ return mn
      else if $r < 2/2exp(-\epsilon/2)$ return mx
      else return $mn + (mx - mn)(r - exp(-\epsilon/2))/(1 - exp(-\epsilon/2))$
    - else if return $q'\left(\frac{sum(D)+L(0,(mx-mn)/\epsilon)}{S}\right)$
    - end if

Differential privacy

- Implementation of clamping-down mean. Applying it to
  - the interval: [2000, 4000]
  - so, sensitivity $\Delta_{\mathcal{D}}(q) = (max - min) = 2000$
  - and the database: (mean = 3300)
    D={1000, 2000, 3000, 2000, 1000, 6000, 2000, 10000, 2000, 4000}
  - Applying the procedure 10000 times, and ploting the histogram



Differential privacy

- Properties of differential privacy
  - On the ϵ:
    - Small ϵ, more privacy, more noise into the solution
    - Large ϵ, less privacy, less noise into the solution
  - On the *sensitivity*:
    - Small sensitivity, less noise for achieving the same privacy
    - Large sensitivity, more noise for achieving the same privacy
  - Discussion here is for a single query (with privacy budget ϵ). Multiple queries (even multiple applications of the same query) need special treatment. E.g., additional privacy budget.
  - Randomness via e.g. Laplace means that any number can be selected. Including e.g. negative ones for salaries. Special treatment may be necessary.
  - Implementations for other type of functions
    - The exponential mechanism for non-numerical queries
    - Differential privacy for machine learning and statistical models

Centralized approach: trusted third party

Trusted third party

Computation-driven approaches/multiple databases: centralized

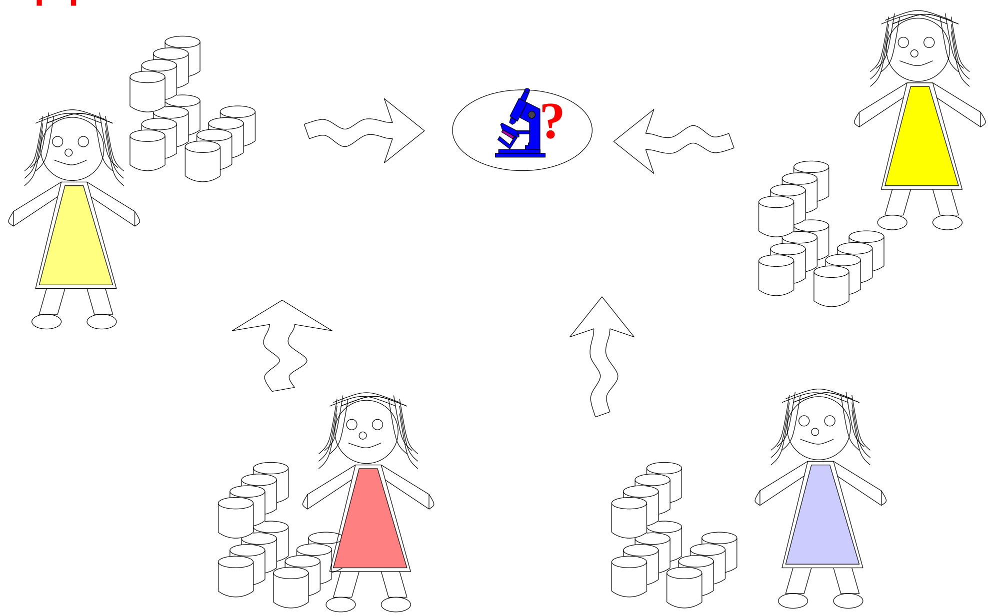
- **Example.** Parties $P_1, \ldots, P_n$ own databases $DB_1, \ldots, DB_n$. The parties want to compute a function, say f, of these databases (i.e., $f(DB_1, \ldots, DB_n)$) without revealing unnecessary information. In other words, after computing $f(DB_1, \ldots, DB_n)$ and delivering this result to all P_i, what P_i knows is nothing more than what can be deduced from his DB_i and the function f.
- So, the computation of f has not given P_i any extra knowledge.

Distributed approach: secure multiparty computation

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

- The centralized approach as a reference



Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Compute the sum of salaries of 4 people: Aine, Brianna, Cathleen, and Deirdre.
  We denote these salaries by s_1, s_2, s_3, and s_4, respectively.
- Each person's salary is confidential and they do not want to share.
- Define a protocol to compute involving only the 4 people (no trusted third party).
- Assume that the sum lies in the range $[0, n]$.

□ Example with 4 people. Similar method applies with other number of people.

□ We use public-key cryptography. I.e., each party requires two separate keys: a private and a public one. This is also known as asymmetric cryptography.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \; mod \; n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \; mod \; n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \; mod \; n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \; mod \; n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \; mod \; n$) to Aine encrypted with Aine's public key.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0, n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \; mod \; n$ will be a number uniformly distributed in $[0, n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \; mod \; n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 \; mod \; n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \; mod \; n$) to Aine encrypted with Aine's public key.
- Aine decrypts Deirdre's message with Aine's private key. She substracts (modulo n) the random number r added in the first step, obtaining in this way $s_1 + s_2 + s_3 + s_4$ (this will be in $[0, n]$).

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Aine adds a secret random number, say r (uniformly chosen in $[0,n]$) to her salary and sends it to Brianna encrypted with Brianna public key. Addition is modulo n. In this way, the outcome of $r + s_1 \ mod \ n$ will be a number uniformly distributed in $[0,n]$ and so Brianna will learn nothing about the actual value of s_1.
- Brianna decrypts Aine's message with Brianna's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 \ mod \ n$) to Cathleen encrypted with Cathleen's public key.
- Cathleen decrypts Brianna's message with Cathleen's private key, adds her salary (modulo n) and sends the result (i.e., $r+s_1+s_2+s_3 \ mod \ n$) to Deirdre encrypted with Deirdre's public key.
- Deirdre decrypts Cathleen's message with Deirdre's private key, adds her salary (modulo n) and sends the result (i.e., $r + s_1 + s_2 + s_3 + s_4 \ mod \ n$) to Aine encrypted with Aine's public key.
- Aine decrypts Deirdre's message with Aine's private key. She substracts (modulo n) the random number r added in the first step, obtaining in this way $s_1+s_2+s_3+s_4$ (this will be in $[0,n]$).
- Aine announces the result to the participants.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- This protocol assumes that all of the participants are honest
- A participant can lie about her salary.
- Aine can announce a wrong addition.
- Participants can collude. E.g.,
  - Brianna and Deirdree can share their figures to find the salary of Cathleen

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- Solving collusion.
  - Each salary is divided into shares.
  - The sum of each share is computed individually.
  - Different paths are used for different shares in a way that neighbors are different.

    To compute any s_i all neighbors of all paths are required.
  - Different number of shares imply different minimum coalition sizes for violating security

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

Important observation

- This method is compliant with the privacy model selected: Secure multiparty computation
- This method is not compliant with other privacy models: differential privacy

We can define appropriate methods that satisfy multiple privacy models

- E.g., method that computes differentially private secure sum

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed. Sum

- We can also apply Shamir's secret sharing approach to this problem

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed.

- Yao's millionaire problem. Alice and Bob want to know who is richer, but they do not want to tell the other how much money they have. This is the secure computation of $a > b$.
- Secure set union.
- Scalar product. Alice with vector x and Bob with vector y want to compute xy.

Secure multiparty computation

Computation-driven approaches/multiple databases/distributed.

- Dining Cryptographers Problem.
  - (Chaum, 1985) Three cryptographers are sitting down to dinner at their favorite three-star restaurant. Their waiter informs them that arrangements have been made with the maître d'hôtel for the bill to be paid anonymously. One of the cryptographers might be paying the dinner, or it might have been NSA (U.S. National Security Agency). The three cryptographers respect each other's right to make an anonymous payment, but they wonder if NSA is paying.
- This problem (and previous ones) can be seen from a user's privacy perspective (more particularly, about protecting the data of the user). I.e., the cryptographers does not want to share whether they paid or not.

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed.

- Machine learning and data mining methods.
- Parties can be seen as sharing the schema of a database.
- Two types of problems usually considered.
  - Vertically partitioned data. Parties (data holders) have information on the same individuals but different attributes.
  - Horizontally partitioned data. Parties (data holders) have information on different individuals but on the same attributes (i.e., the share the database schema).

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

Privacy leakage for the distributed approach is usually analyzed considering two types of adversaries.

Secure multiparty computation

Computation-driven approaches/multiple databases: distributed

Privacy leakage for the distributed approach is usually analyzed considering two types of adversaries.

- **Semi-honest adversaries.** Data owners follow the cryptographic protocol but they analyse all the information they get during its execution to discover as much information as they can.
- **Malicious adversaries.** Data owners try to fool the protocol (e.g. aborting it or sending incorrect messages on purpose) so that they can infer confidential information.