Differential privacy without a central database

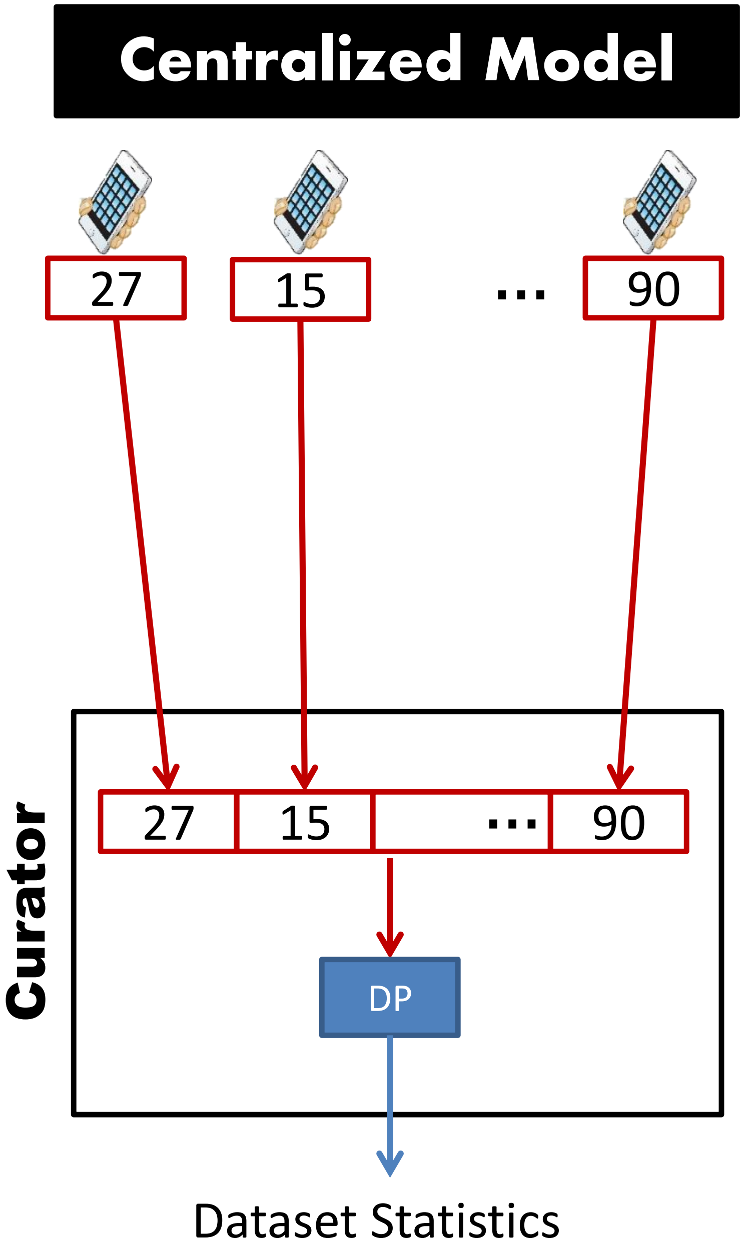
Boston Differential Privacy Summer School, 6-10 June 2022

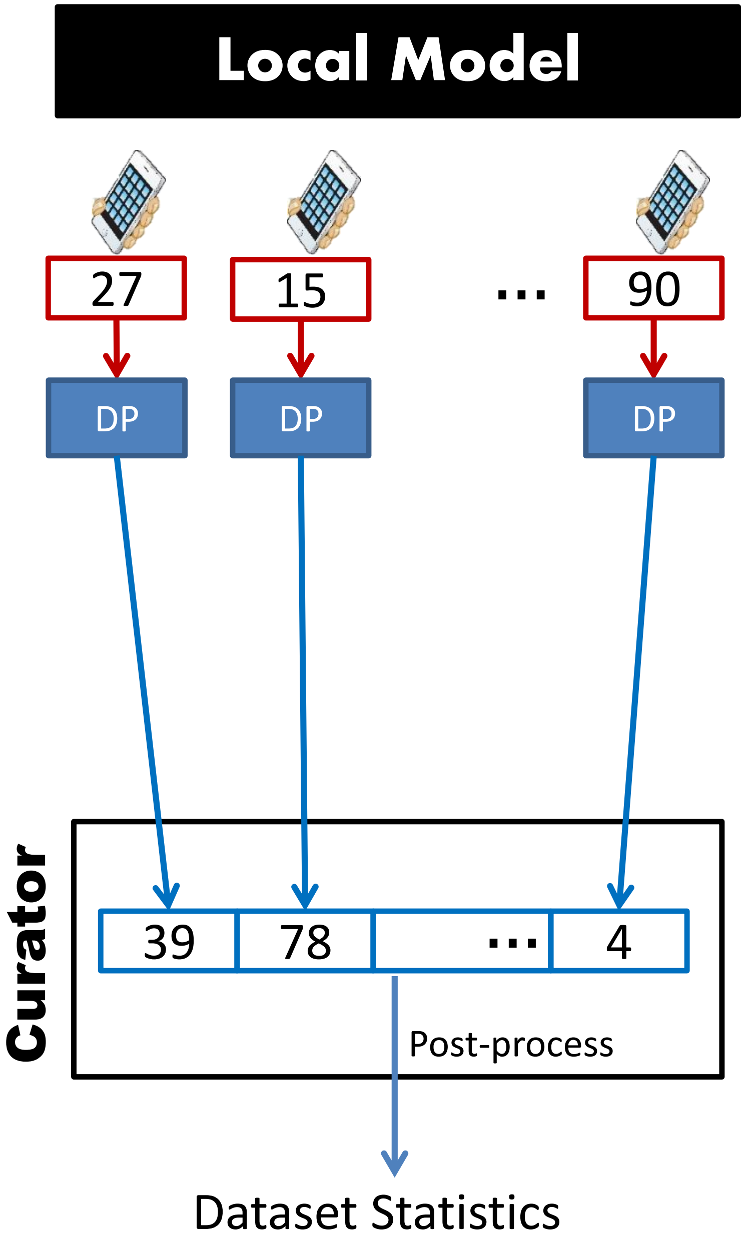
Uri Stemmer

About this course

- The local model ✓
- The shuffle model
- Streaming/online settings
- Differential privacy as a tool

Last time: Local Differential Privacy





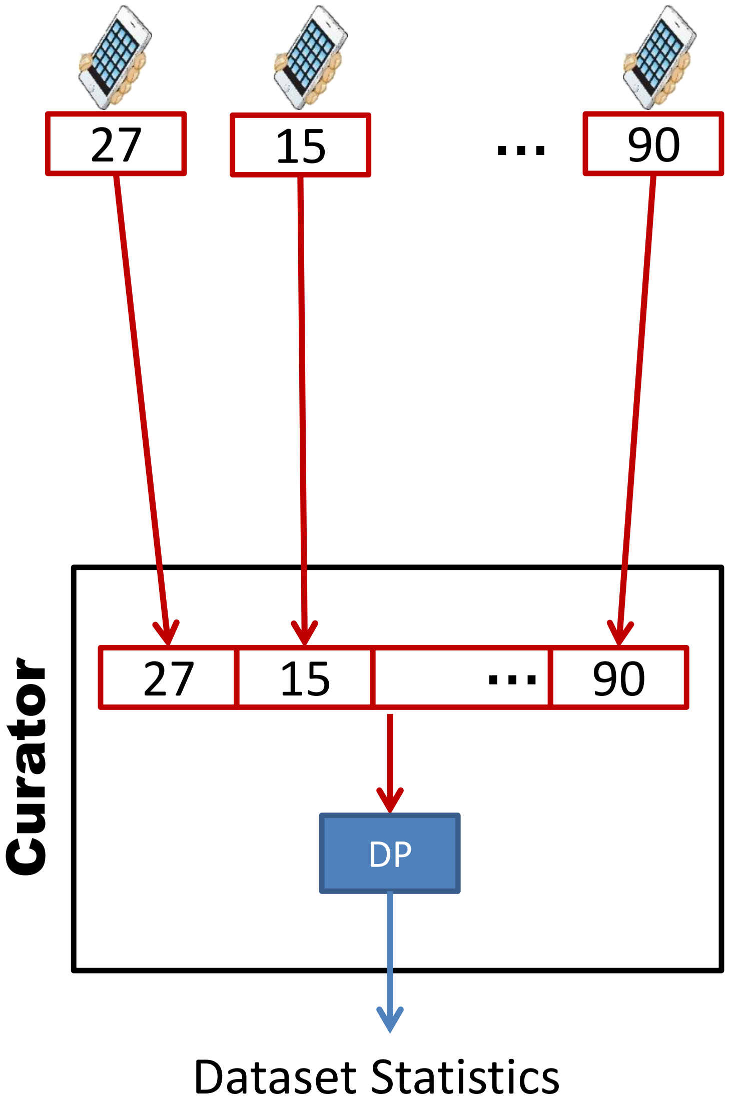
- Users retain their data
- Only send randomizations which are safe for publication

✓ No need to trust anyone

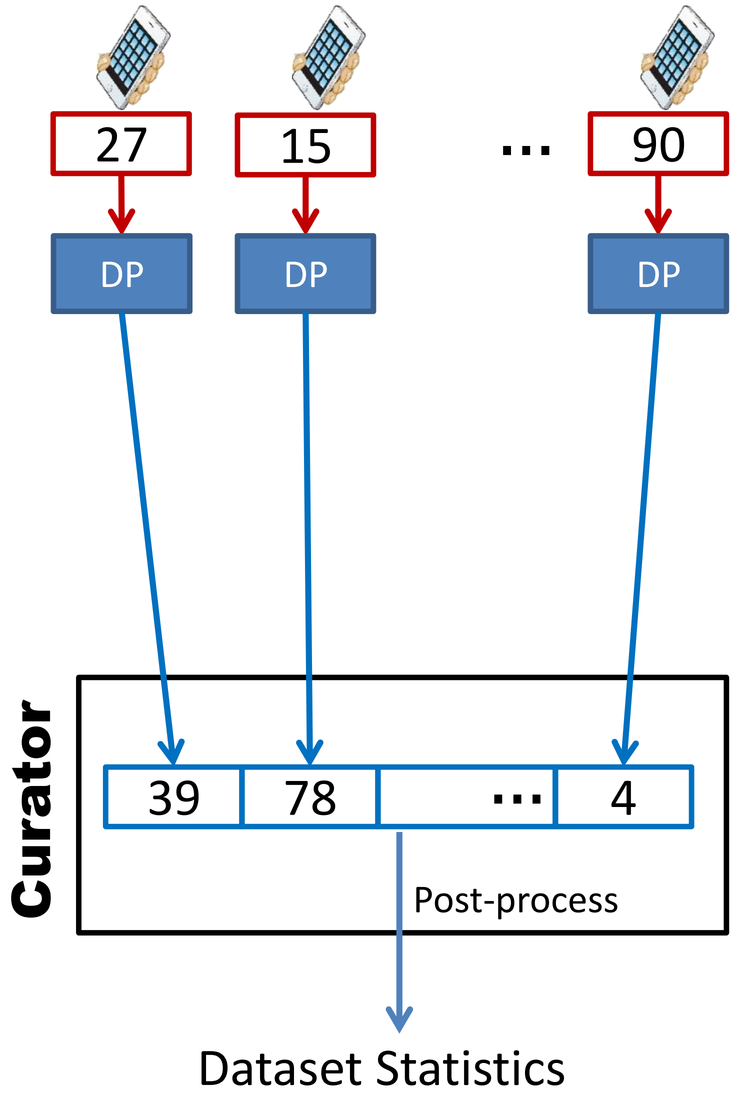
✗ Accuracy is reduced

Last time: Local Differential Privacy

Centralized Model



Local Model



- Users retain their data
- Only send randomizations which are safe for publication

✓ No need to trust anyone

× Accuracy is reduced

The natural question:
Can we get the best of both worlds?

The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

- Alice and Bob wants to decide whether or not to get married
- Bob doesn’t know what Alice wants, and if she says no he will be embarrassed
- Same with Alice

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

- Alice and Bob wants to decide whether or not to get married
- Bob doesn't know what Alice wants, and if she says no he will be embarrassed
- Same with Alice

Goal: Design a process in which Alice and Bob learn if there is mutual love, and nothing else

Notice: If Alice loves Bob then at the end of the process she learns whether Bob loves her or not. We want that if Alice does not love Bob, then at the end of the process she will not learn Bob's answer

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

Here's what we'll need:

- A coin
- 6 cards with ones and zeroes: 0 1 0 1 0 1

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

Here's what we'll need:

- A coin
- 6 cards with ones and zeroes:

0	1	0	1	0	1

What we are going to do:

- Alice and bob will "shuffle" the cards on the table (the cards are faced down on the table)
- At the and of the process they will learn if there is mutual love

This question makes sense also without DP (and has been studied way before DP…):

Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

Here's what we'll need:

- A coin
- 6 cards with ones and zeroes: 0 1 0 1 0 1

What we are going to do:

- Alice and bob will "shuffle" the cards on the table (the cards are faced down on the table)
- At the and of the process they will learn if there is mutual love

Notations: 0 1 $\Longrightarrow$ Encode the answer "no"

1 0 $\Longrightarrow$ Encode the answer "yes"

This question makes sense also without DP (and has been studied way before DP…):

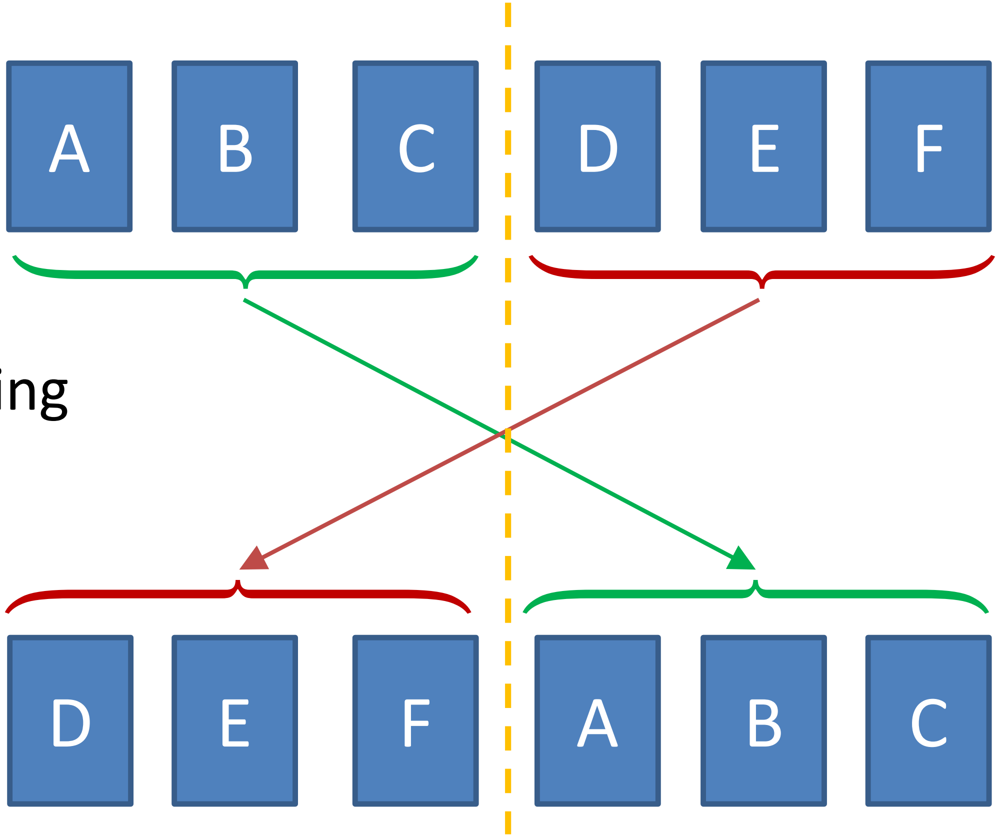
Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

Example: The Wedding Problem

Let's define a "random swap" operation:

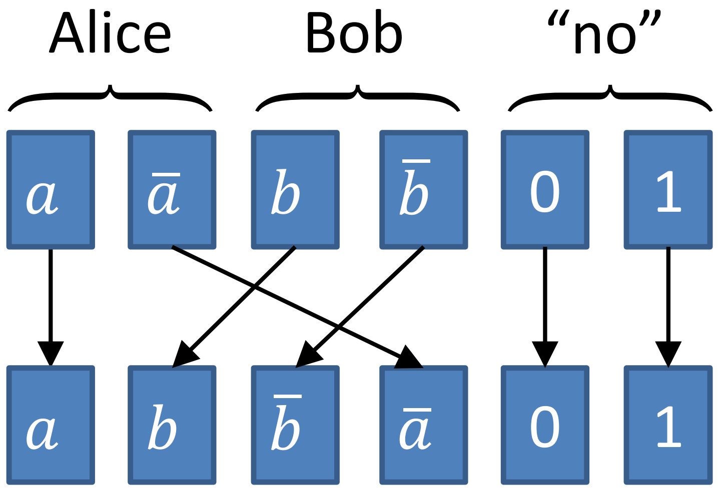
- Begin with 6 cards:



- Toss a coin
- Heads we change nothing
- Tails we swap:

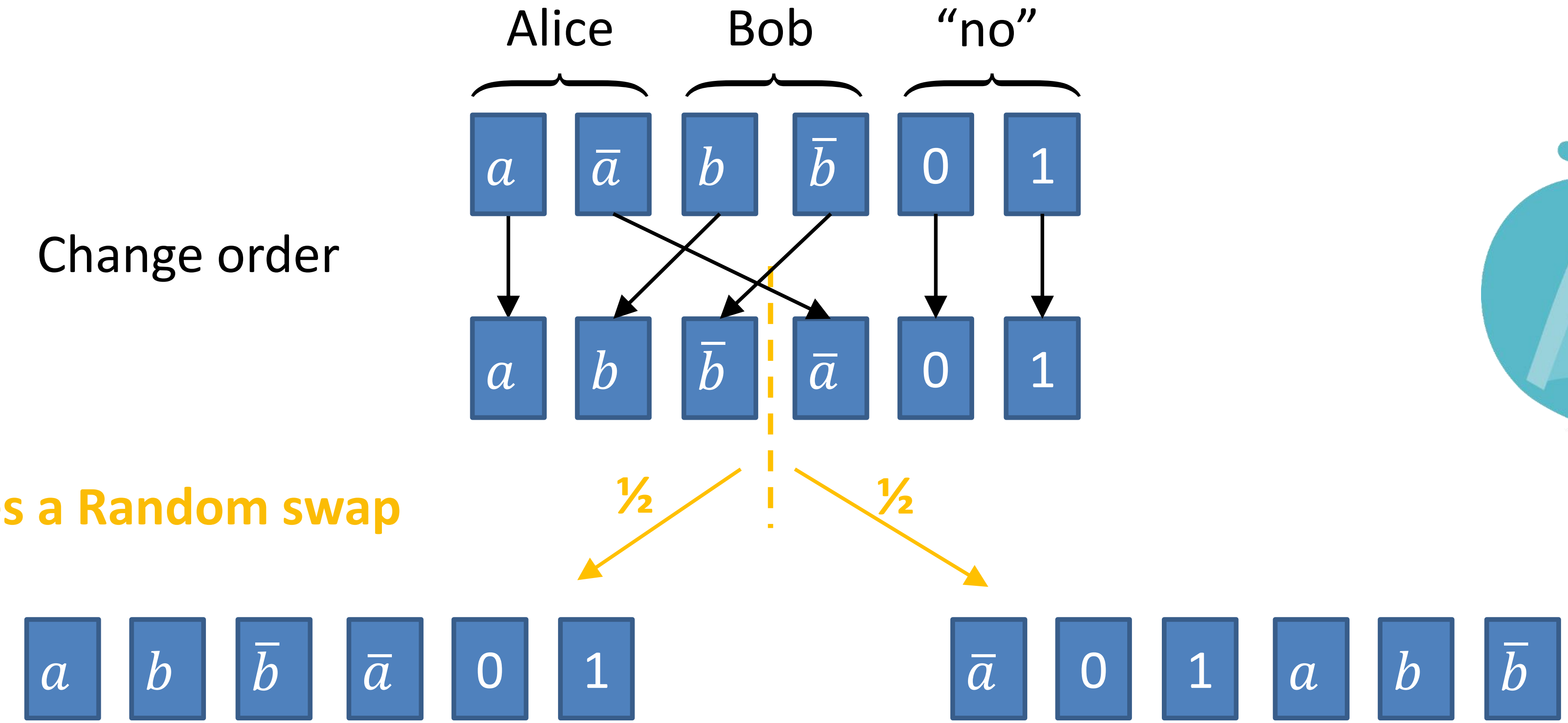
$$\overbrace{\boxed{a}\quad\boxed{\bar{a}}}^{\text{Alice}}\quad\overbrace{\boxed{b}\quad\boxed{\bar{b}}}^{\text{Bob}}\quad\overbrace{\boxed{0}\quad\boxed{1}}^{\text{“no”}}$$

Change order

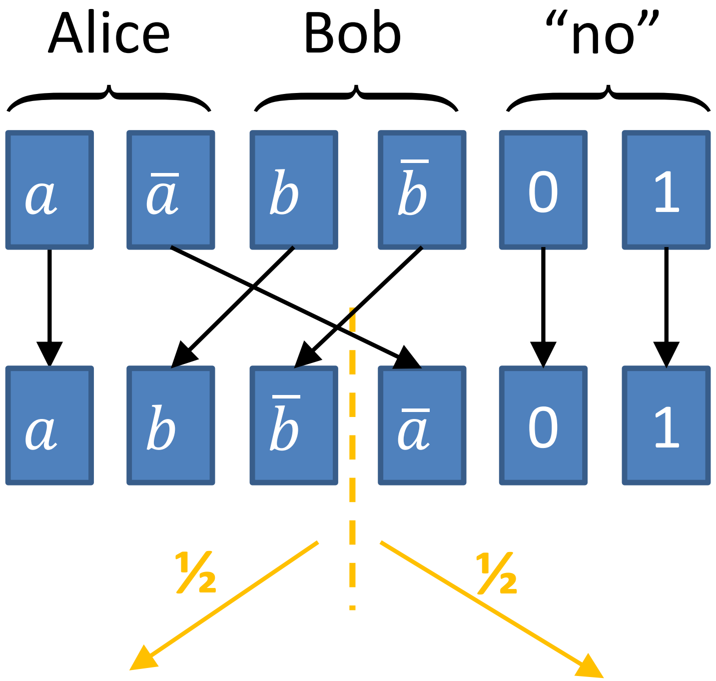


Change order

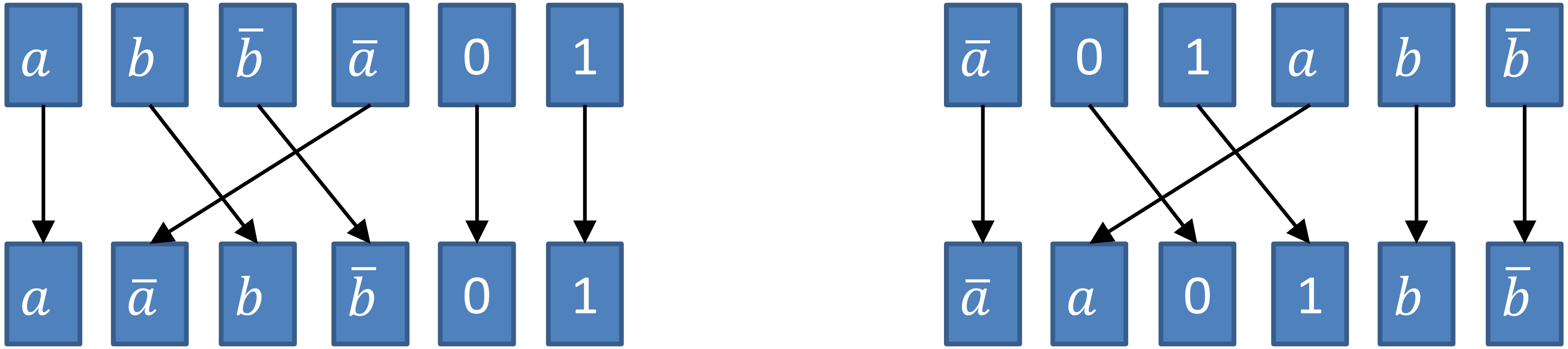
Alice does a Random swap



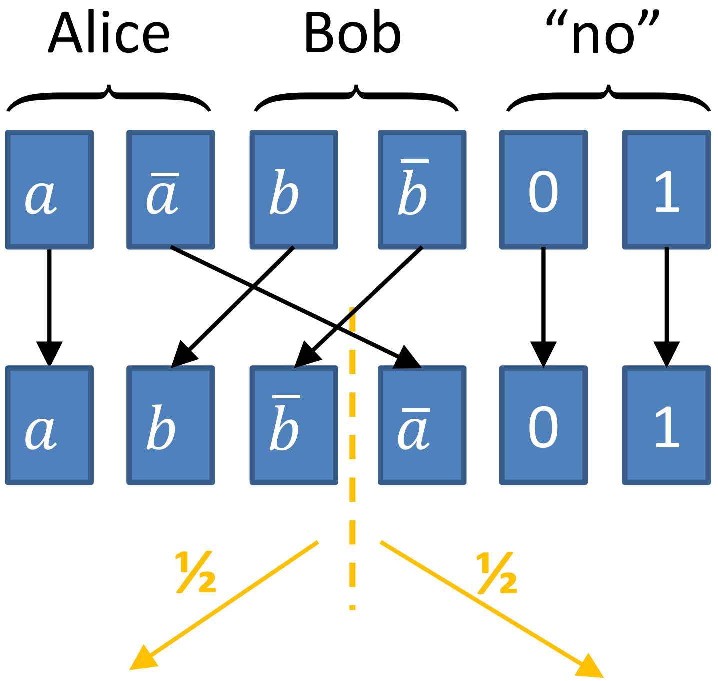
Change order



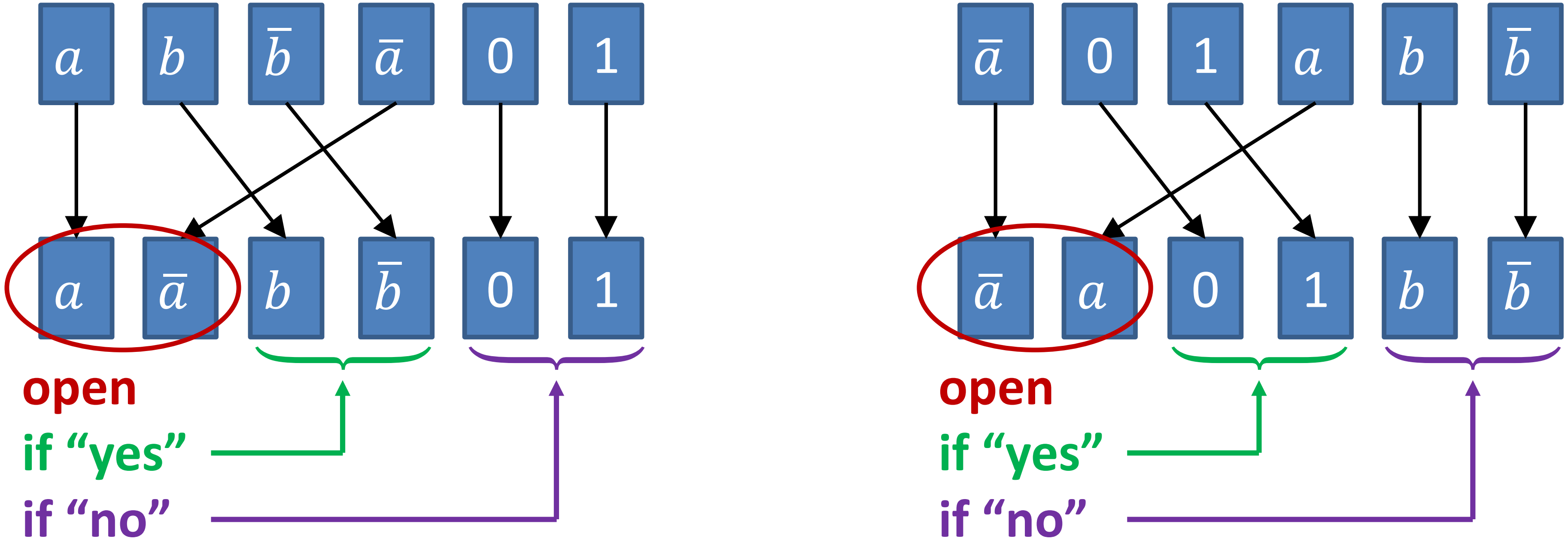
Alice does a Random swap



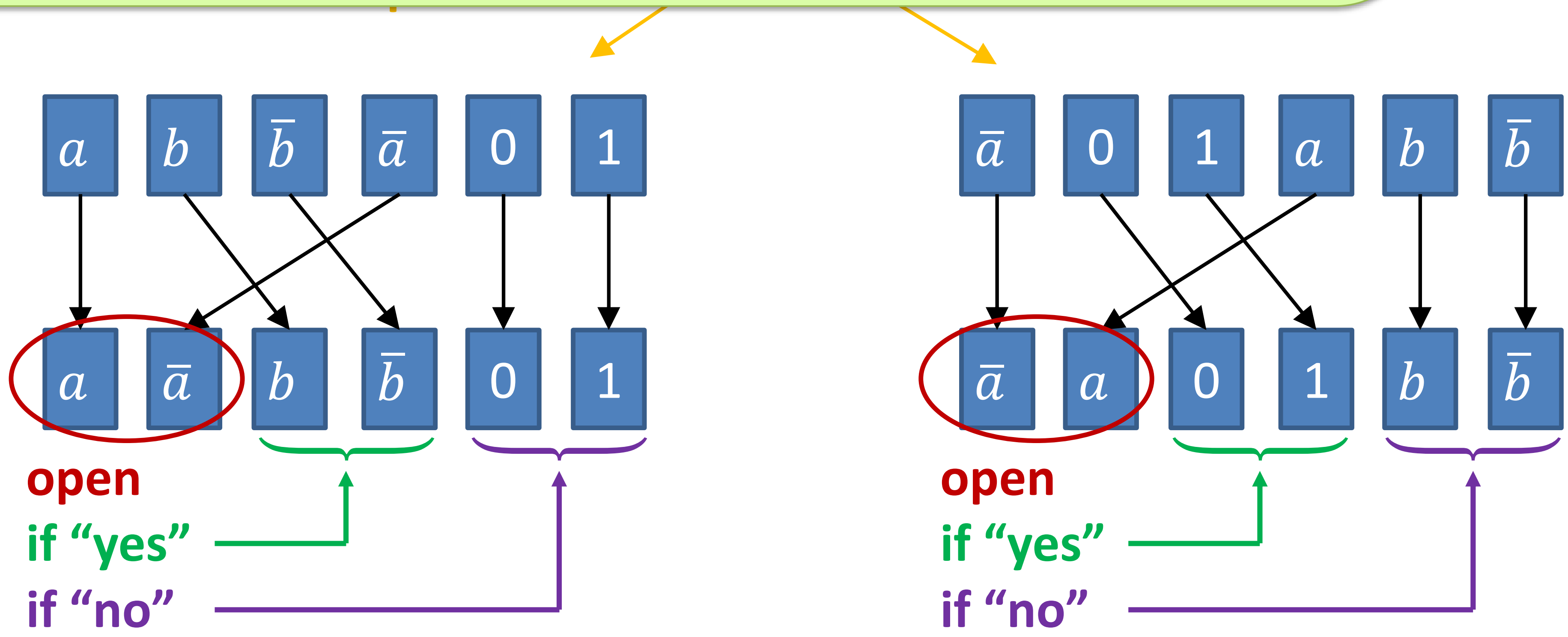
Change order



Alice does a Random swap



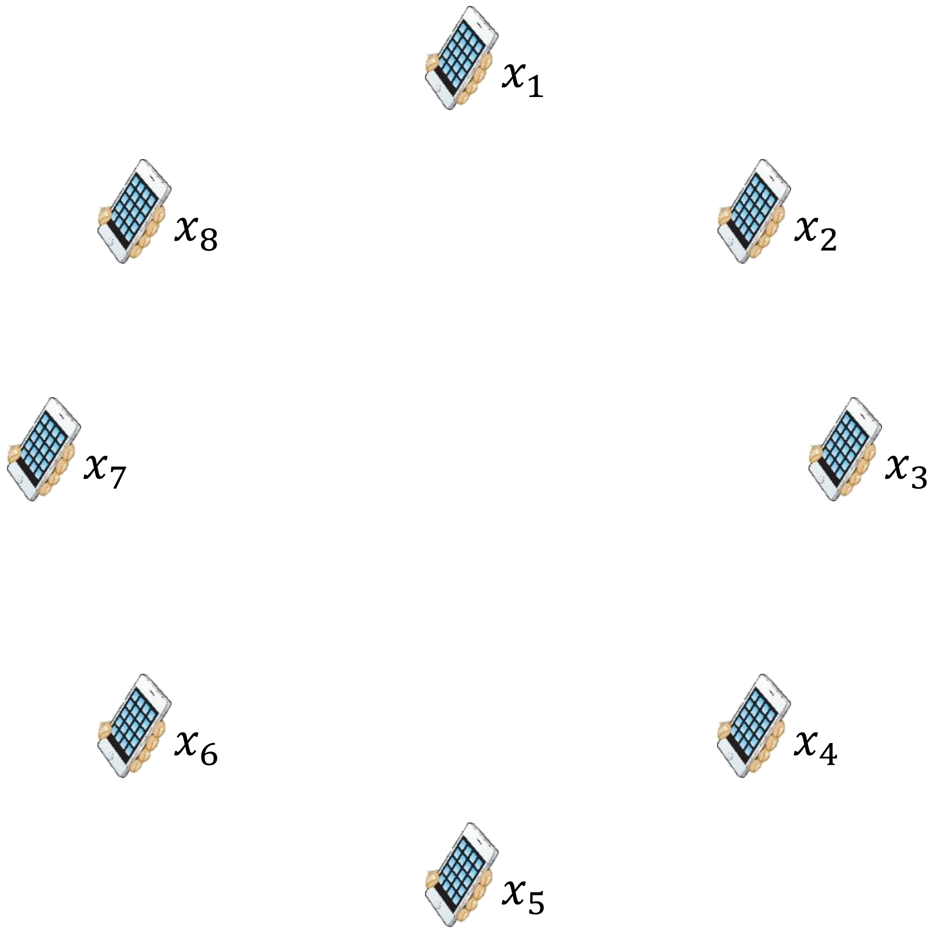
- If Alice is a “no” then Bob’s cards are never opened, and Alice learns nothing
- In Bob’s eyes, Alice’s cards are randomly swapped. If Bob is a “no” then the other 4 cards are 0101 so it doesn’t matter what we open and Bob learns nothing
- This is a simple example for secure 2-party computation for the function AND



Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

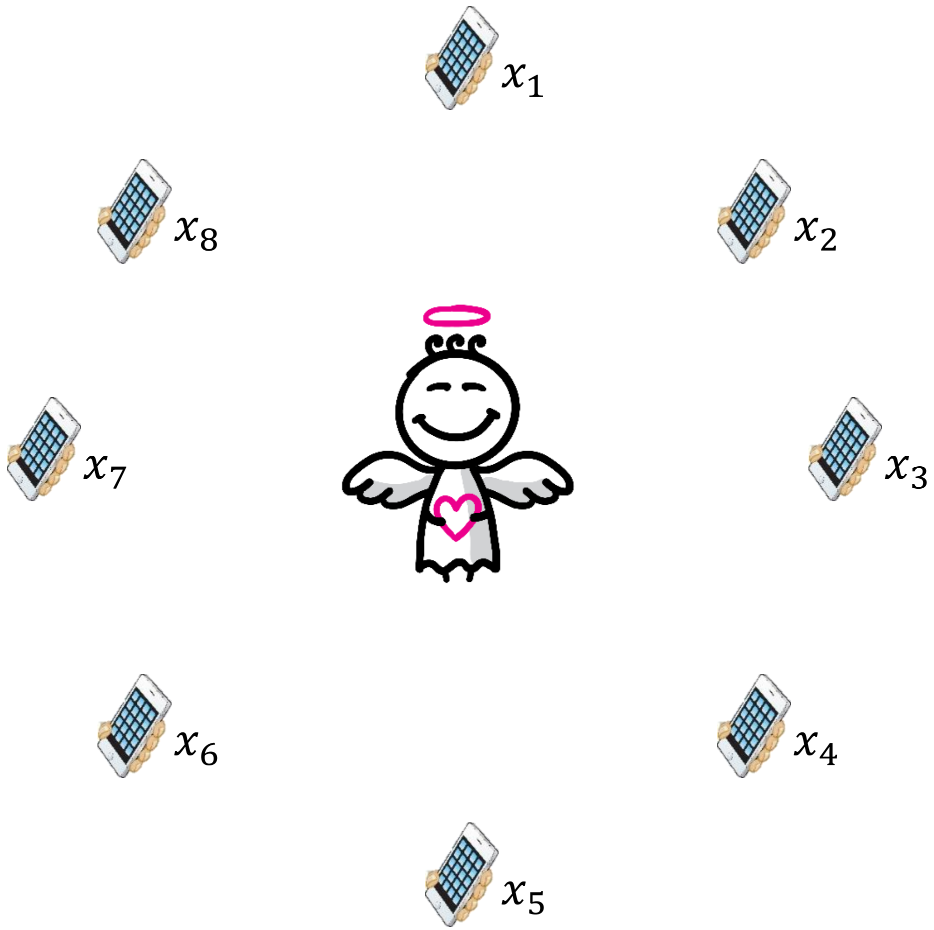
- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \dots, P_n$ holding inputs $x_1, \dots, x_n$
- The players want to compute $f(x_1, \dots, x_n)$ without revealing anything more



Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

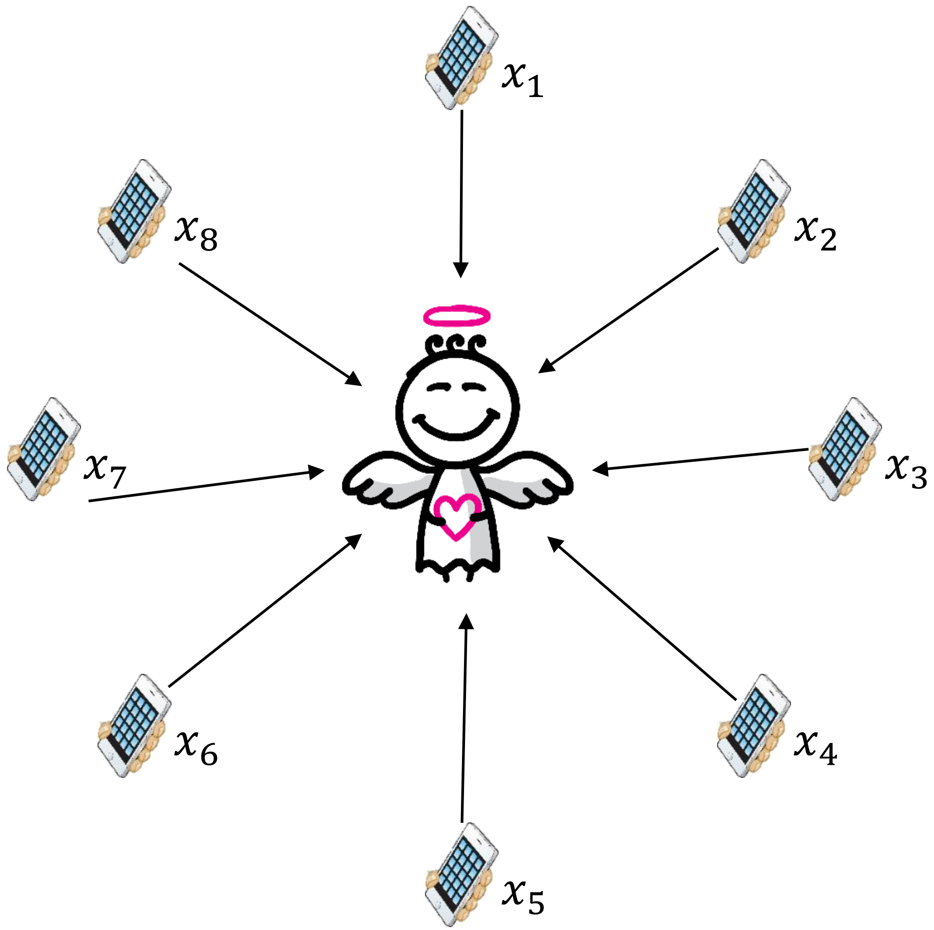
- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \ldots, P_n$ holding inputs $x_1, \ldots, x_n$
- The players want to compute $f(x_1, \ldots, x_n)$ without revealing anything more



Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

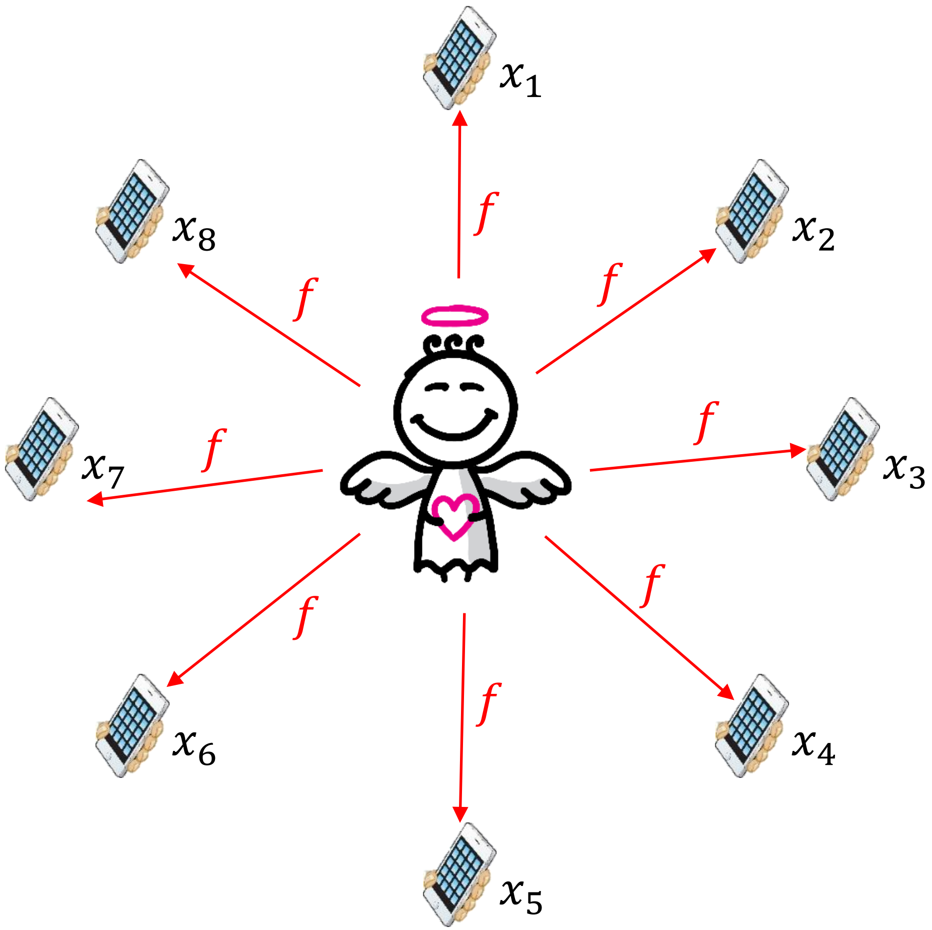
- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \ldots, P_n$ holding inputs $x_1, \ldots, x_n$
- The players want to compute $f(x_1, \ldots, x_n)$ without revealing anything more



Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

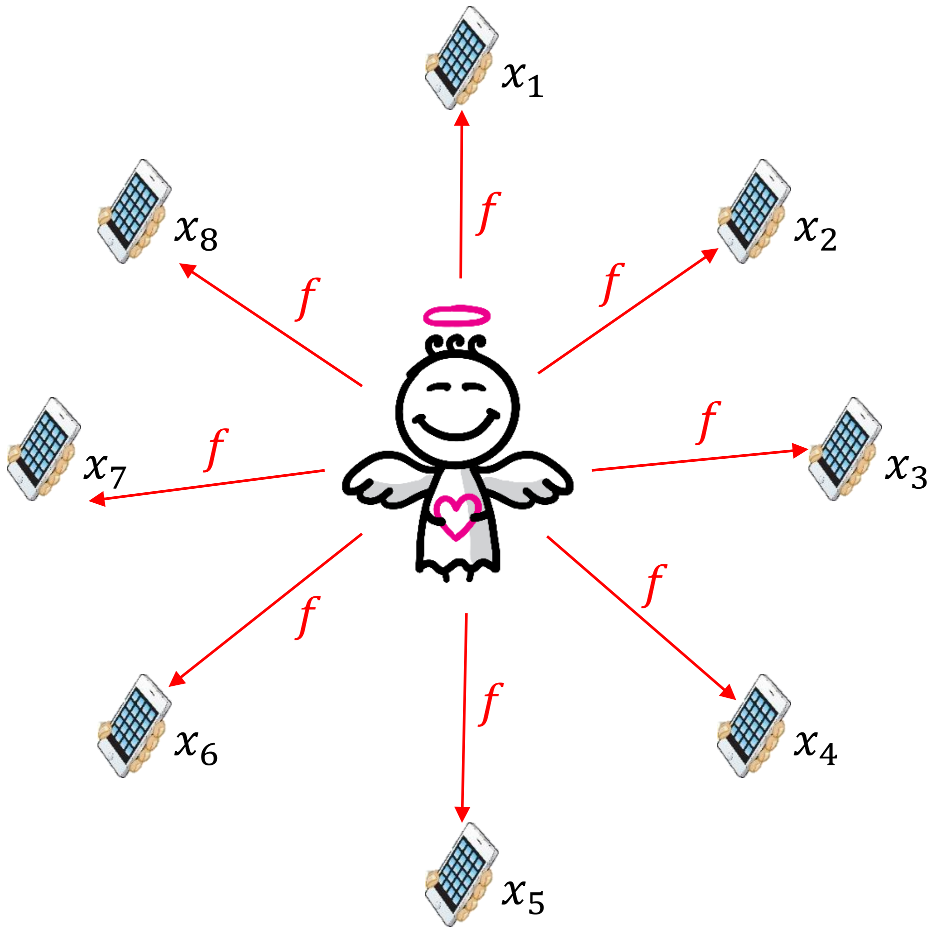
- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \ldots, P_n$ holding inputs $x_1, \ldots, x_n$
- The players want to compute $f(x_1, \ldots, x_n)$ without revealing anything more

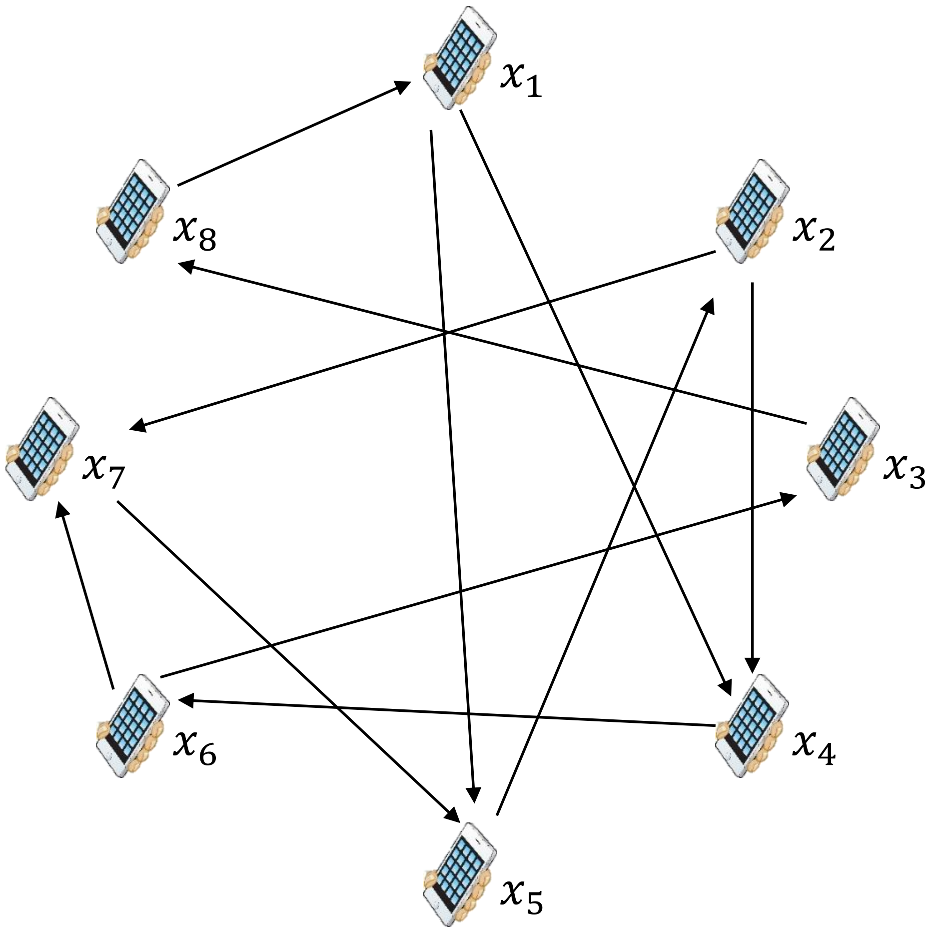


Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \dots, P_n$ holding inputs $x_1, \dots, x_n$
- The players want to compute $f(x_1, \dots, x_n)$ without revealing anything more

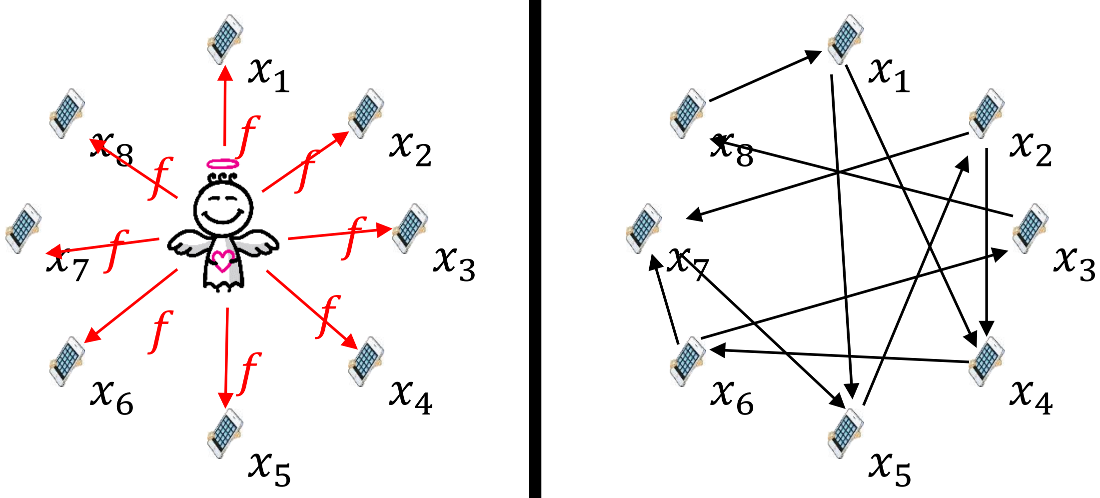




Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \dots, P_n$ holding inputs $x_1, \dots, x_n$
- The players want to compute $f(x_1, \dots, x_n)$ without revealing anything more



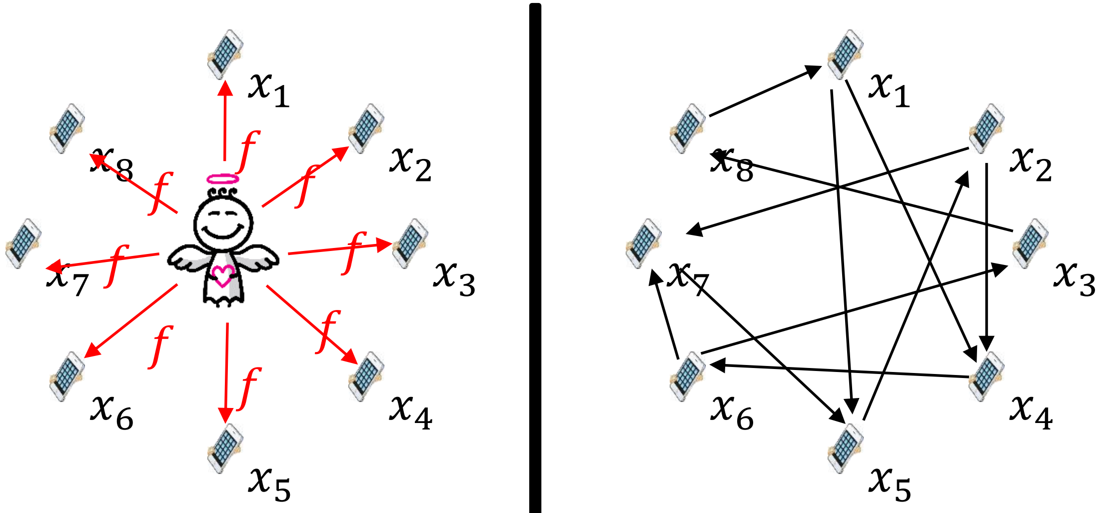
Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \dots, P_n$ holding inputs $x_1, \dots, x_n$
- The players want to compute $f(x_1, \dots, x_n)$ without revealing anything more

- Informally, a protocol for f is <u>secure</u> if it emulates the ideal world in the sense that any adversary (controlling a subset of the parties) cannot learn anything more than what it can learn in the ideal world

- Many different settings: How many parties can the adversary control? Adaptive vs static corruptions? Semi-honest vs malicious? Poly-time or computationally unbounded adversary? Communication network?



Secure Multiparty Computation (MPC)

[Yao] [Goldreich, Micali, Wigderson]

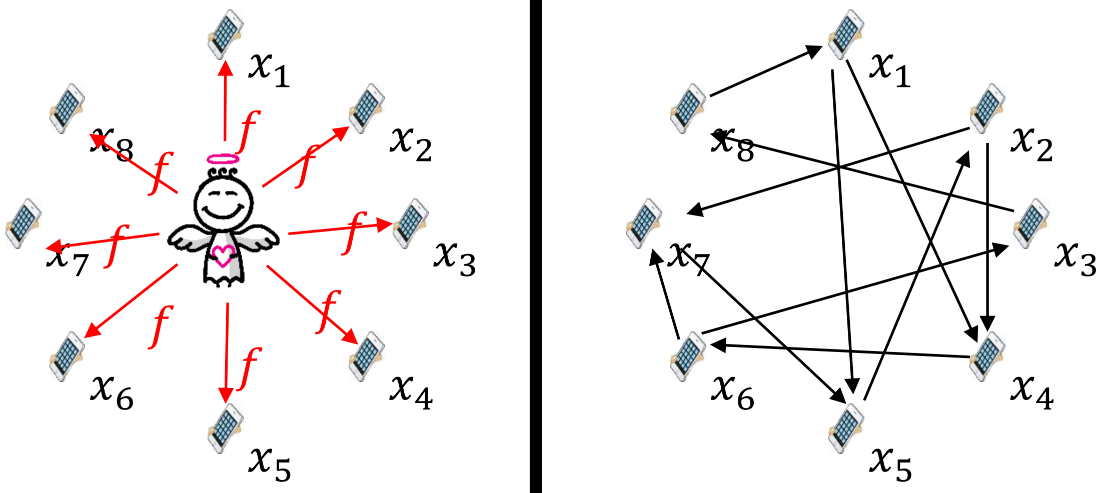
- Let $f: X^n \to Y$ be an n-input function (possibly randomized)
- n players $P_1, \dots, P_n$ holding inputs $x_1, \dots, x_n$
- The players want to compute $f(x_1, \dots, x_n)$ without revealing anything more

- Informally, a protocol for f is <u>secure</u> if it emulates the ideal world in the sense that any adversary (controlling a subset of the parties) cannot learn anything more than what it can learn in the ideal world

- Many different settings: How many parties can the adversary control? Adaptive vs static corruptions? Semi-honest vs malicious? Poly-time or computationally unbounded adversary? Communication network?

<u>Informal theorem:</u> Secure multiparty can be achieved for any function f assuming less than a third of the parties can be corrupted [Goldreich, Micali, Wigderson 87] [Ben-Or, Goldwasser, Wigderson 88]

<u>Remark :</u> This only means that we know HOW to compute f, Not that it is necessarily a good idea in terms of privacy…

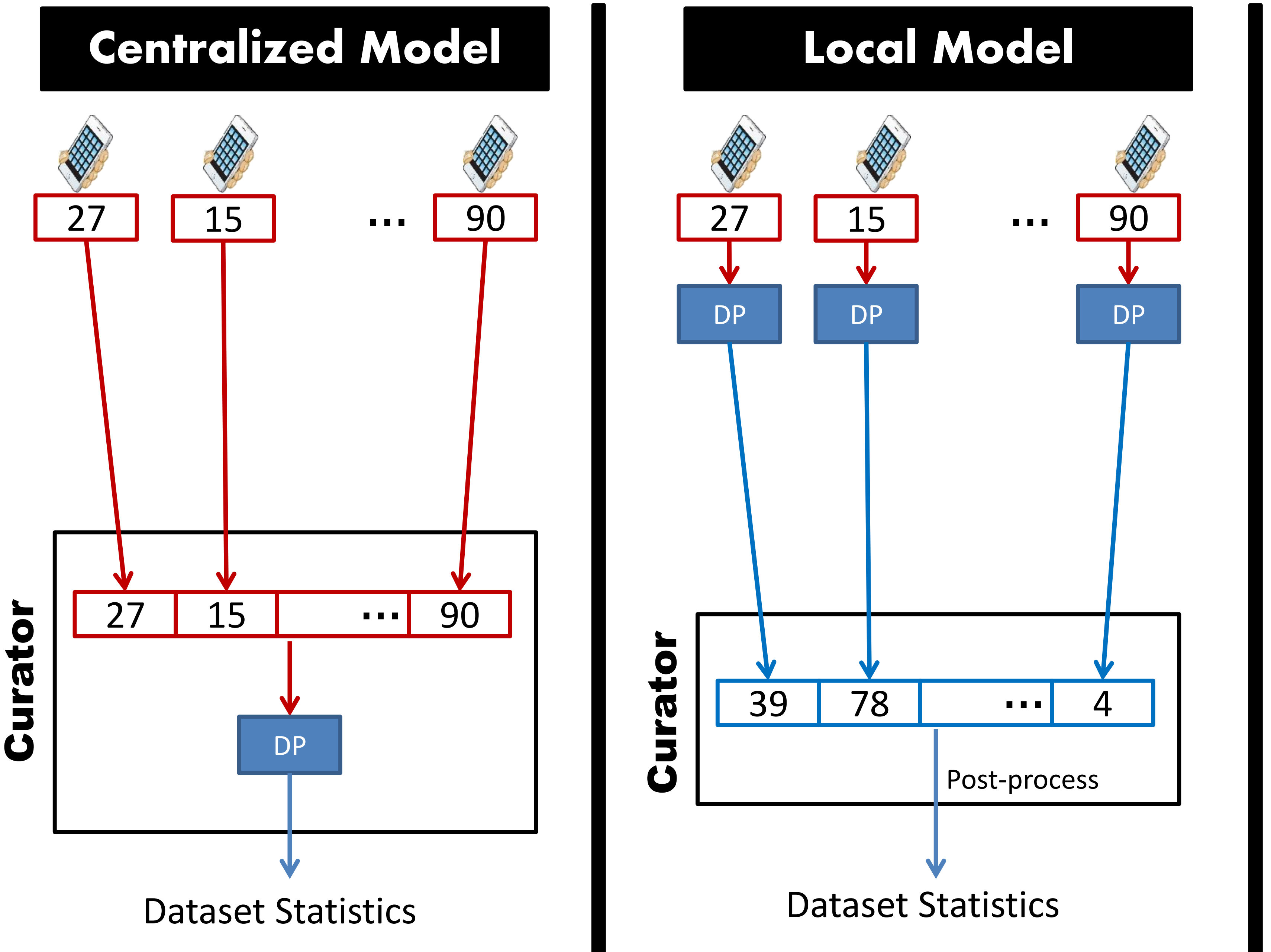


The Shuffle Model of Differential Privacy

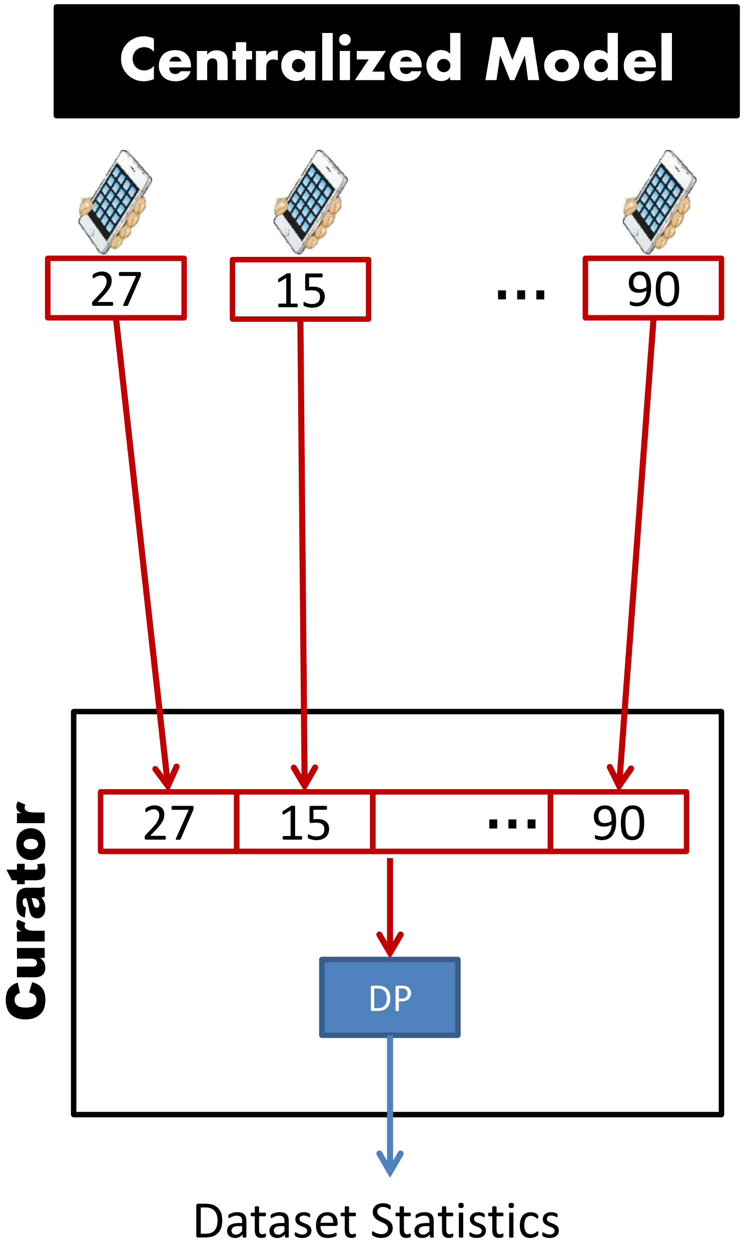
Today's Outline

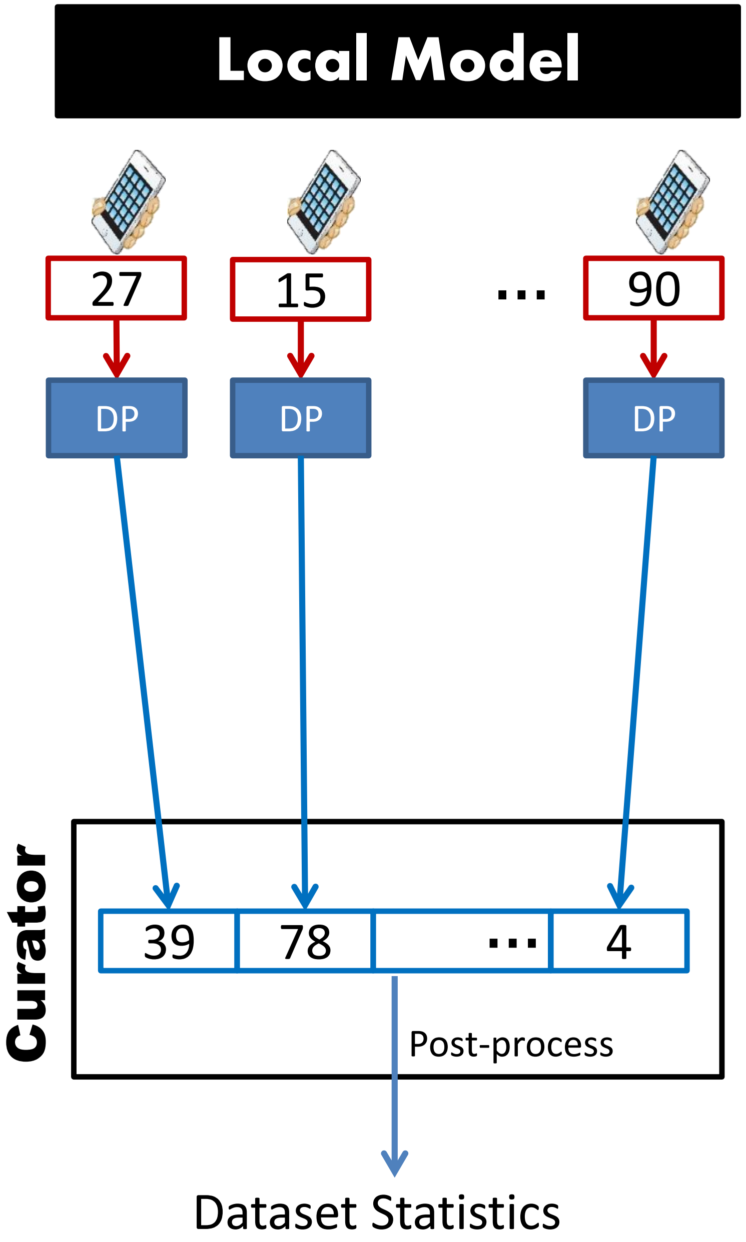
1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

Back to our context



Back to our context



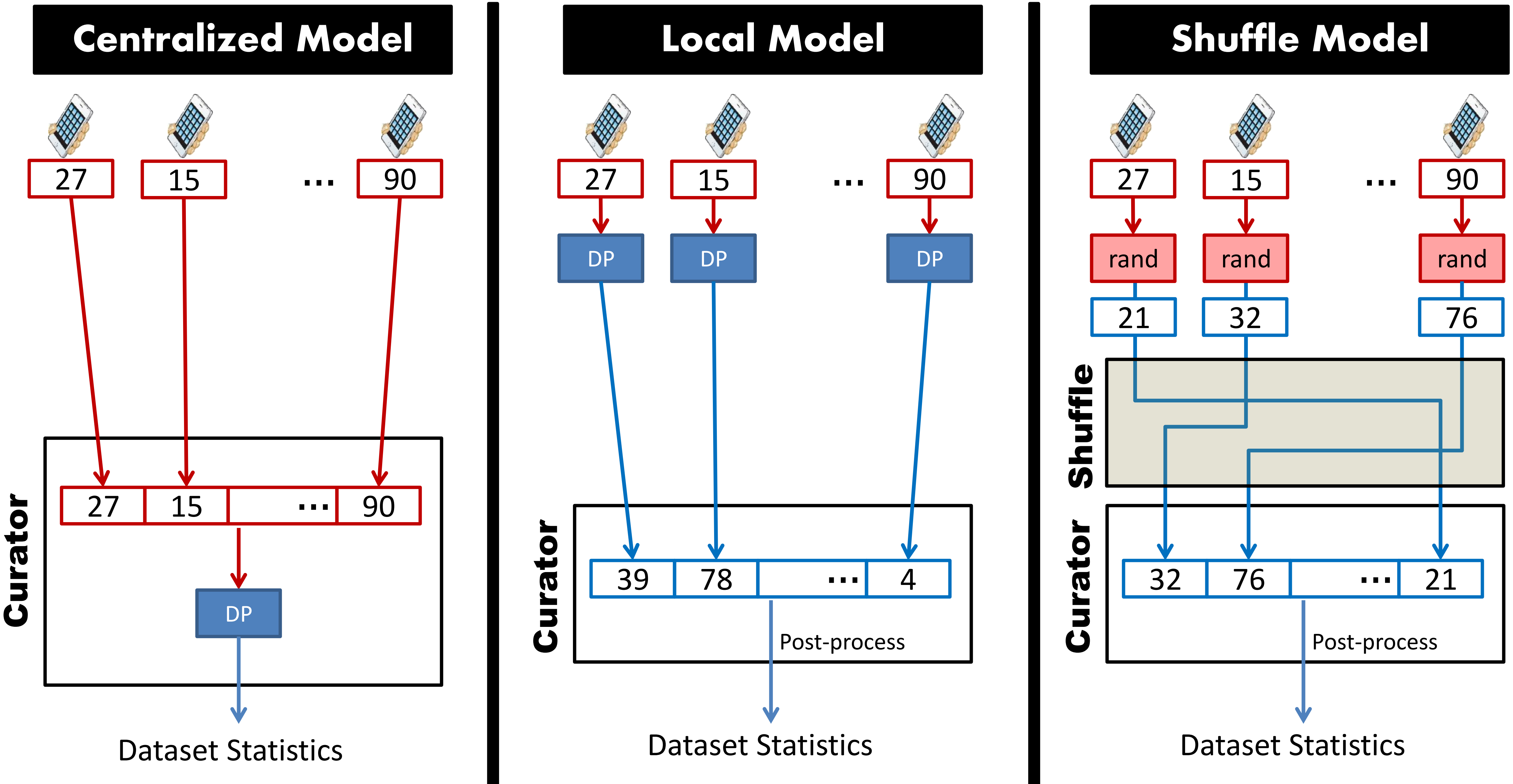


Applying MPC to a DP functionality f in the centralized model, we get a protocol for computing f without a trusted entity!

The downside is that generic MPC constructions are generally quite complex, requiring several rounds of communication with large overhead

The shuffle model

Bittau, Erlingsson, Maniatis, Mironov, Raghunathan, Lie, Rudominer, Kode, Tinnes, Seefeld 2017
Erlingsson, Feldman, Mironov, Raghunathan, Talwar, Thakurta 2019
Cheu, Smith, Ullman, Zeber, Zhilyaev 2019

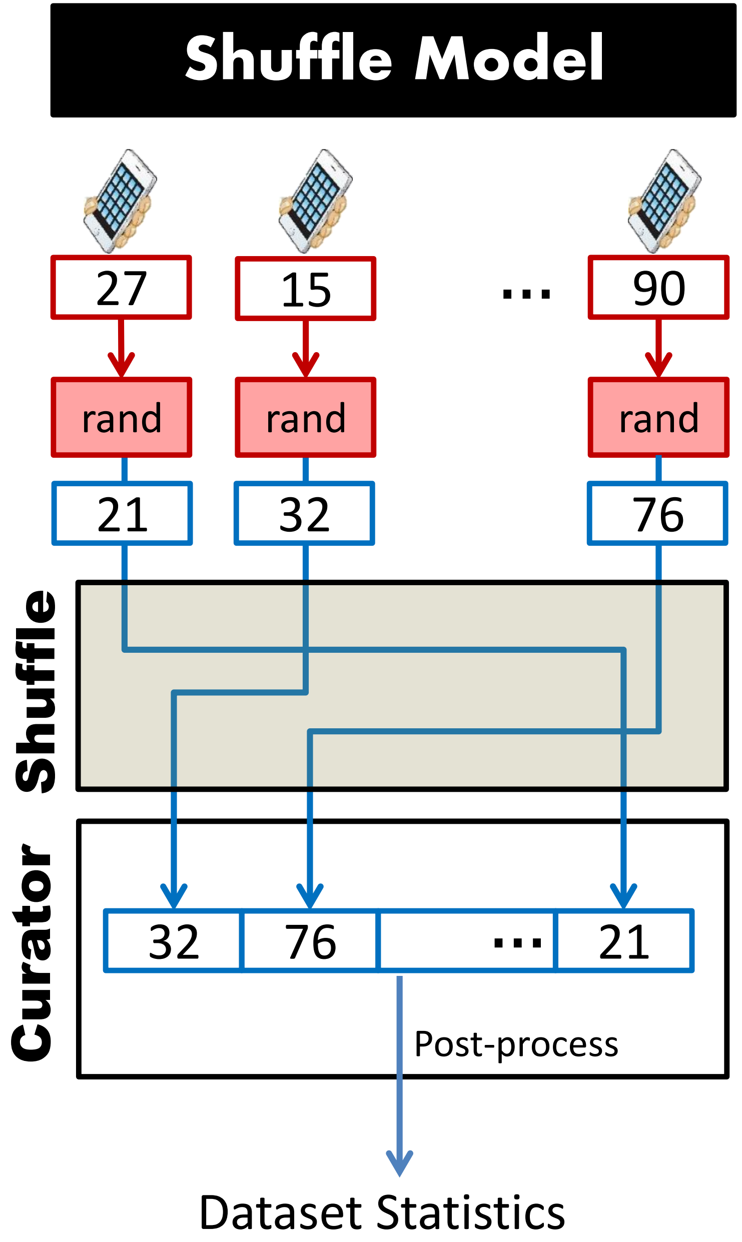


The shuffle model

Bittau, Erlingsson, Maniatis, Mironov, Raghunathan, Lie, Rudominer, Kode, Tinnes, Seefeld 2017
Erlingsson, Feldman, Mironov, Raghunathan, Talwar, Thakurta 2019
Cheu, Smith, Ullman, Zeber, Zhilyaev 2019

Definition:

- There are $\boldsymbol{n}$ users and a server
- Each user $\boldsymbol{i}$ holds an input $\boldsymbol{x_i} \in \boldsymbol{X}$
- Each user $\boldsymbol{i}$ runs (locally) a randomization algorithm $\boldsymbol{R}$ to obtain ℓ messages: $(\boldsymbol{m}_{i,1}, \dots, \boldsymbol{m}_{i,\ell}) \leftarrow \boldsymbol{R}(\boldsymbol{x}_i)$
- The users submit these messages to a special communication channel called *shuffle*
- At the outcome of the shuffle we get a random permutation of the $\boldsymbol{n}\ell$ messages, denoted as $\textbf{Shuffle}(\boldsymbol{m}_{1,1}, \dots, \boldsymbol{m}_{1,\ell}, \dots, \boldsymbol{m}_{n,1}, \dots, \boldsymbol{m}_{n,\ell})$
- The server post-processes the outcome of the shuffle



The shuffle model

Bittau, Erlingsson, Maniatis, Mironov, Raghunathan, Lie, Rudominer, Kode, Tinnes, Seefeld 2017
Erlingsson, Feldman, Mironov, Raghunathan, Talwar, Thakurta 2019
Cheu, Smith, Ullman, Zeber, Zhilyaev 2019

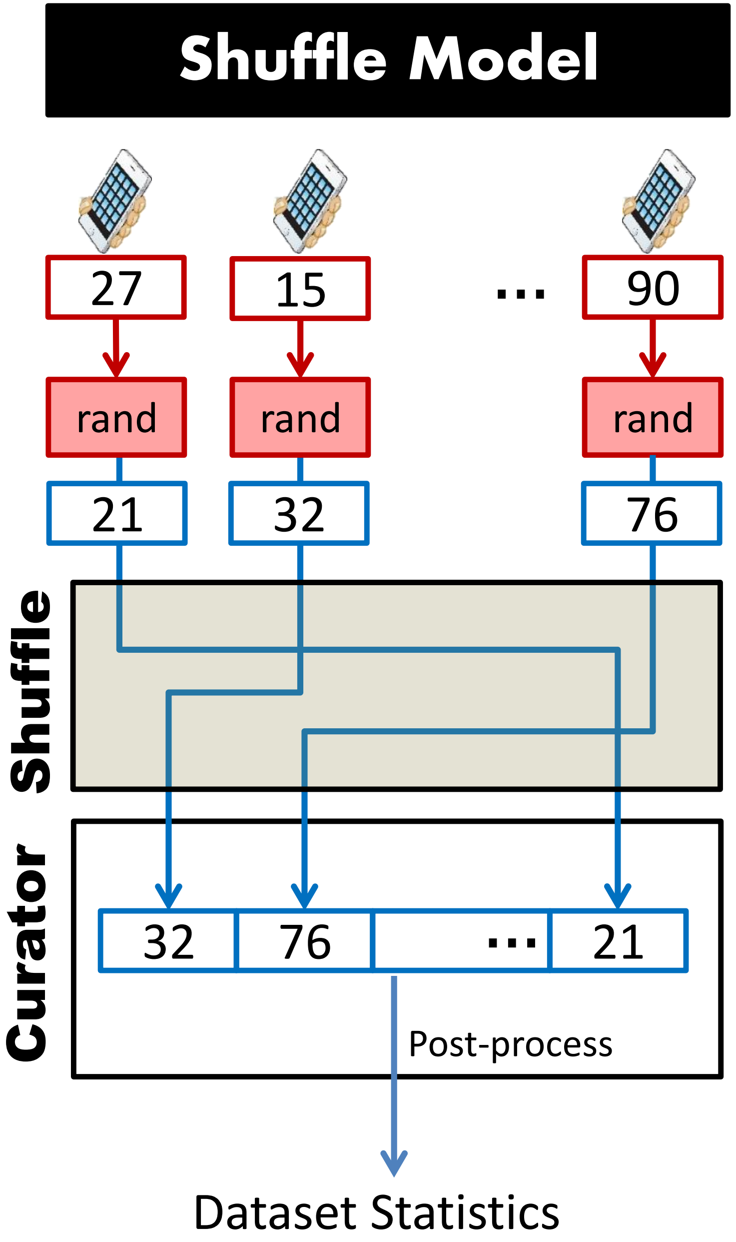
Definition:

- There are $\boldsymbol{n}$ users and a server
- Each user $\boldsymbol{i}$ holds an input $\boldsymbol{x_i} \in \boldsymbol{X}$
- Each user $\boldsymbol{i}$ runs (locally) a randomization algorithm $\boldsymbol{R}$ to obtain ℓ messages: $(\boldsymbol{m_{i,1}}, \dots, \boldsymbol{m_{i,\ell}}) \leftarrow \boldsymbol{R}(\boldsymbol{x_i})$
- The users submit these messages to a special communication channel called *shuffle*
- At the outcome of the shuffle we get a random permutation of the $\boldsymbol{n\ell}$ messages, denoted as $\mathbf{Shuffle}(\boldsymbol{m_{1,1}}, \dots, \boldsymbol{m_{1,\ell}}, \dots, \boldsymbol{m_{n,1}}, \dots, \boldsymbol{m_{n,\ell}})$
- The server post-processes the outcome of the shuffle

Privacy requirement at the outcome of the shuffle:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$ and any event $\boldsymbol{F}$ we have

$$\Pr[\text{shuffle}(R(x_1), \dots, R(x_n)) \in F] \leq e^{\varepsilon} \cdot \Pr[\text{shuffle}(R(x'_1), \dots, R(x'_n)) \in F] + \delta$$



The shuffle model

Bittau, Erlingsson, Maniatis, Mironov, Raghunathan, Lie, Rudominer, Kode, Tinnes, Seefeld 2017
Erlingsson, Feldman, Mironov, Raghunathan, Talwar, Thakurta 2019
Cheu, Smith, Ullman, Zeber, Zhilyaev 2019

Definition:

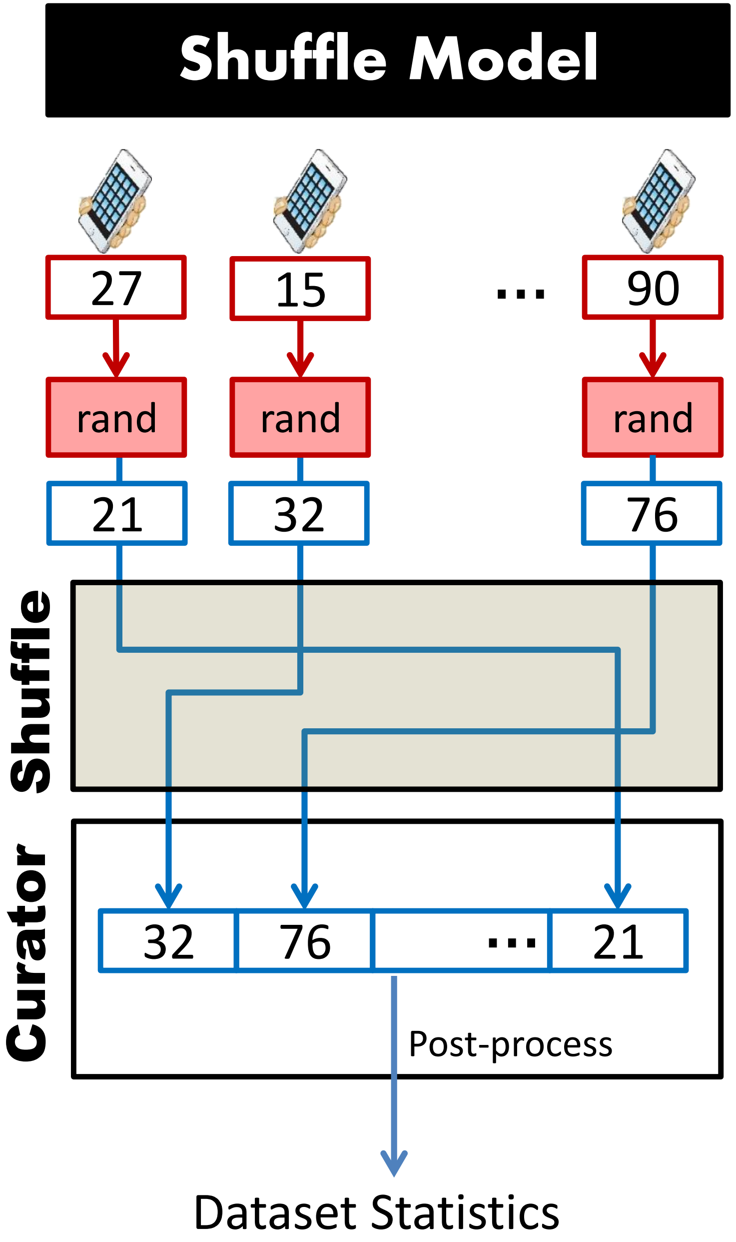
- There are $\boldsymbol{n}$ users and a server
- Each user $\boldsymbol{i}$ holds an input $\boldsymbol{x_i} \in \boldsymbol{X}$
- Each user $\boldsymbol{i}$ runs (locally) a randomization algorithm $\boldsymbol{R}$ to obtain ℓ messages: $(\boldsymbol{m_{i,1}}, \dots, \boldsymbol{m_{i,\ell}}) \leftarrow \boldsymbol{R}(\boldsymbol{x_i})$
- The users submit these messages to a special communication channel called *shuffle*
- At the outcome of the shuffle we get a random permutation of the $\boldsymbol{n\ell}$ messages, denoted as $\mathbf{Shuffle}(\boldsymbol{m_{1,1}}, \dots, \boldsymbol{m_{1,\ell}}, \dots, \boldsymbol{m_{n,1}}, \dots, \boldsymbol{m_{n,\ell}})$
- The server post-processes the outcome of the shuffle

Privacy requirement at the outcome of the shuffle:

For any neighboring datasets $\vec{x}, \vec{x}'$ and any event $\boldsymbol{F}$ we have

$$\Pr[\text{shuffle}(R(x_1), \dots, R(x_n)) \in F] \le e^{\varepsilon} \cdot \Pr[\text{shuffle}(R(x_1'), \dots, R(x_n')) \in F] + \delta$$

Simple observations: (1) Shuffle model is no stronger than the centralized model since the curator can simulate it; **(2)** LDP is no stronger than shuffle



The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

Counting bits in the shuffle model

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Theorem: Can count bits with error $O\left(\frac{1}{\varepsilon}\right)$

% Recall the $\Omega\left(\frac{\sqrt{n}}{\varepsilon}\right)$ error in the local model

% We show only $\approx \frac{1}{\varepsilon}\log\frac{1}{\delta}$ and assume for simplicity that $n \gg \frac{1}{\varepsilon^2}\ln\left(\frac{1}{\delta}\right)$

Counting bits in the shuffle model

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Theorem: Can count bits with error $O\left(\frac{1}{\varepsilon}\right)$

% Recall the $\Omega\left(\frac{\sqrt{n}}{\varepsilon}\right)$ error in the local model

% We show only $\approx \frac{1}{\varepsilon}\log\frac{1}{\delta}$ and assume for simplicity that $n \gg \frac{1}{\varepsilon^2}\ln\left(\frac{1}{\delta}\right)$

Algorithm R (on the users' side): Input $x \in \{0,1\}$	**Algorithm on the server's side:** Input $b_1, \dots, b_{2n} \in \{0,1\}$
(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$ (2) Return $m_1 = x \quad , \quad m_2 = y$	(1) Return $\left(\sum_{i=1}^{2n} b_i\right) - np$

Counting bits in the shuffle model

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Theorem: Can count bits with error $O\left(\frac{1}{\varepsilon}\right)$

% Recall the $\Omega\left(\frac{\sqrt{n}}{\varepsilon}\right)$ error in the local model

% We show only $\approx \frac{1}{\varepsilon}\log\frac{1}{\delta}$ and assume for simplicity that $n \gg \frac{1}{\varepsilon^2}\ln\left(\frac{1}{\delta}\right)$

Algorithm R (on the users' side): Input $x \in \{0,1\}$	**Algorithm on the server's side:** Input $b_1, \dots, b_{2n} \in \{0,1\}$
(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$ (2) Return $m_1 = x \quad , \quad m_2 = y$	(1) Return $\left(\sum_{i=1}^{2n} b_i\right) - np$

Utility analysis:

- First observe $\sum_{i=1}^{2n} b_i = \left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} y_i\right) := \left(\sum_{i=1}^{n} x_i\right) + Z$
- Now, by the Chernoff bound, with high probability we have $Z \approx np \pm \sqrt{np}$
- Which gives us an error of roughly $\frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)$

Counting bits in the shuffle model

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Theorem: Can count bits with error $O\left(\frac{1}{\varepsilon}\right)$

% Recall the $\Omega\left(\frac{\sqrt{n}}{\varepsilon}\right)$ error in the local model

% We show only $\approx \frac{1}{\varepsilon}\log\frac{1}{\delta}$ and assume for simplicity that $n \gg \frac{1}{\varepsilon^2}\ln\left(\frac{1}{\delta}\right)$

Algorithm R (on the users' side): Input $x \in \{0,1\}$	**Algorithm on the server's side:** Input $b_1, \dots, b_{2n} \in \{0,1\}$
(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$ (2) Return $m_1 = x \quad , \quad m_2 = y$	(1) Return $\left(\sum_{i=1}^{2n} b_i\right) - np$

Utility analysis:

- First observe $\sum_{i=1}^{2n} b_i = \left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} y_i\right) := \left(\sum_{i=1}^{n} x_i\right) + Z$
- Now, by the Chernoff bound, with high probability we have $Z \approx np \pm \sqrt{np}$
- Which gives us an error of roughly $\frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)$

Privacy Analysis:

- Suffices to show that the sum $B := \sum_{i=1}^{2n} b_i$ satisfies DP, because the outcome of the shuffle is determined by this (random permutation of B ones and $2n - B$ zeroes)

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x \quad , \quad m_2 = y$

- Suffices to show DP for $B := \sum_{i=1}^{2n} b_i$
- We denote $Z := \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} := I_{\mathrm{Good}}$

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x \quad , \quad m_2 = y$

- Suffices to show DP for $B := \sum_{i=1}^{2n} b_i$
- We denote $Z := \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} := I_{\mathrm{Good}}$

Let $\boldsymbol{k} \in \boldsymbol{np} \pm \sqrt{\boldsymbol{np}}$ be a "likely" value for $\boldsymbol{Z}$. We have

$$\frac{\Pr[Z = k]}{\Pr[Z = k - 1]}$$

$$\lesssim e^{\varepsilon}$$

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x$, $m_2 = y$

- Suffices to show DP for $B := \sum_{i=1}^{2n} b_i$
- We denote $Z := \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} := I_{\mathrm{Good}}$

Let $k \in np \pm \sqrt{np}$ be a "likely" value for Z. We have

$$\frac{\Pr[Z=k]}{\Pr[Z=k-1]} = \frac{\binom{n}{k} \cdot p^k \cdot (1-p)^{n-k}}{\binom{n}{k-1} \cdot p^{k-1} \cdot (1-p)^{n-k+1}} = \frac{\frac{n!}{k! \cdot (n-k)!}}{\frac{n!}{(k-1)! \cdot (n-k+1)!}} \cdot \frac{p}{(1-p)} = \frac{n-k+1}{k} \cdot \frac{p}{1-p}$$

$$= \frac{n-k+1}{k} \cdot \frac{pn}{n-pn} = \frac{n-k+1}{n-pn} \cdot \frac{pn}{k} \le \left(1 + \frac{\sqrt{np}+1}{n-pn}\right) \cdot \frac{pn}{pn - \sqrt{np}}$$

$$\lesssim \left(1 + \sqrt{\frac{p}{n}}\right) \cdot \frac{1}{1 - \sqrt{\frac{1}{pn}}} \approx \left(1 + \frac{1}{\varepsilon n}\right) \cdot \frac{1}{1-\varepsilon} \le (1+\varepsilon) \cdot \frac{1}{1-\varepsilon} \lesssim e^{\varepsilon}$$

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \text{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x$, $m_2 = y$

- Suffices to show DP for $B \coloneqq \sum_{i=1}^{2n} b_i$
- We denote $Z \coloneqq \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} \coloneqq I_{\text{Good}}$
- For $k \in I_{\text{Good}}$ we have $\frac{\Pr[Z=k]}{\Pr[Z=k-1]} \le e^{\varepsilon}$

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \text{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x$, $m_2 = y$

- Suffices to show DP for $B := \sum_{i=1}^{2n} b_i$
- We denote $Z := \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} := I_{\text{Good}}$
- For $k \in I_{\text{Good}}$ we have $\frac{\Pr[Z=k]}{\Pr[Z=k-1]} \leq e^{\varepsilon}$

Now let X be a dataset with t ones, let X' be a neighboring dataset with $t+1$ ones, and let $F \subseteq \mathbb{N}$

$$\Pr\begin{bmatrix} B \in F \\ \text{run on } X \end{bmatrix}$$

$$\leq e^{\varepsilon} \cdot \Pr\begin{bmatrix} B \in F \\ \text{run on } X' \end{bmatrix} + \delta$$

Counting bits - privacy analysis continued

[Cheu, Smith, Ullman, Zeber, Zhilyaev 2019]

Algorithm R (on the users' side): Input $x \in \{0,1\}$

(1) Sample $y \sim \mathrm{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$

(2) Return $m_1 = x \quad , \quad m_2 = y$

- Suffices to show DP for $B \coloneqq \sum_{i=1}^{2n} b_i$
- We denote $Z \coloneqq \sum_{i=1}^{n} y_i$
- by Chernoff, whp: $Z \approx np \pm \sqrt{np} \coloneqq I_{\mathrm{Good}}$
- For $k \in I_{\mathrm{Good}}$ we have $\frac{\Pr[Z=k]}{\Pr[Z=k-1]} \leq e^{\varepsilon}$

Now let X be a dataset with t ones, let X' be a neighboring dataset with $t+1$ ones, and let $F \subseteq \mathbb{N}$

Denote $F_{-t} = \{f - t : f \in F\}$. We have,

$$\Pr\left[\begin{matrix} B \in F \\ \text{run on } X \end{matrix}\right] = \Pr[t + Z \in F] = \Pr[Z \in F_{-t}] = \Pr[Z \in F_{-t} \cap I_{\mathrm{Good}}] + \Pr[Z \in F_{-t} \setminus I_{\mathrm{Good}}]$$

$$\leq \Pr[Z \in F_{-t} \cap I_{\mathrm{Good}}] + \delta \leq e^{\varepsilon} \cdot \Pr[Z + 1 \in F_{-t} \cap I_{\mathrm{Good}}] + \delta$$

$$\leq e^{\varepsilon} \cdot \Pr[Z + 1 \in F_{-t}] + \delta = e^{\varepsilon} \cdot \Pr[Z + 1 + t \in F] + \delta \leq e^{\varepsilon} \cdot \Pr\left[\begin{matrix} B \in F \\ \text{run on } X' \end{matrix}\right] + \delta$$

The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

The Robust Shuffle Model

[Balcer, Cheu, Joseph, Mao]

- The protocol we saw for counting bits has nice feature: It is resilient to dropouts
- Specifically, privacy is still preserved if a constant fraction of the users (say half) do not submit messages to the shuffle

The Robust Shuffle Model

[Balcer, Cheu, Joseph, Mao]

- The protocol we saw for counting bits has nice feature: It is resilient to dropouts
- Specifically, privacy is still preserved if a constant fraction of the users (say half) do not submit messages to the shuffle

Recall the privacy requirement we had before:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$ and any event $\boldsymbol{F}$ we have

$$\Pr\left[\text{shuffle}\left(R(x_1), \dots, R(x_n)\right) \in F\right] \leq e^{\varepsilon} \cdot \Pr\left[\text{shuffle}\left(R(x_1'), \dots, R(x_n')\right) \in F\right] + \delta$$

The Robust Shuffle Model

[Balcer, Cheu, Joseph, Mao]

- The protocol we saw for counting bits has nice feature: It is resilient to dropouts
- Specifically, privacy is still preserved if a constant fraction of the users (say half) do not submit messages to the shuffle

Recall the privacy requirement we had before:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$ and any event $\boldsymbol{F}$ we have

$$\Pr[\text{shuffle}(R(x_1),\ldots,R(x_n)) \in F] \le e^{\varepsilon} \cdot \Pr[\text{shuffle}(R(x'_1),\ldots,R(x'_n)) \in F] + \delta$$

Simple example of a bad protocol:

Algorithm $\boldsymbol{R}$ for user 1: Input $x \in \{0,1\}$ (1) Sample $y_1, y_2, \ldots, y_n \sim \text{Bernouli}(p)$ for $p = \frac{\log\left(\frac{1}{\delta}\right)}{\varepsilon^2 \cdot n}$ (2) Return $m_1 = x$, $m_2 = y_1$, … , $m_{n+1} = y_n$	**Algorithm on the server's side:** Input $b_1, \ldots, b_{2n} \in \{0,1\}$ (1) Return $\left(\sum_{i=1}^{2n} b_i\right) - np$
Algorithm $\boldsymbol{R}$ for users 2-n: Input $x \in \{0,1\}$ (1) Return $m_1 = x$	

The Robust Shuffle Model

[Balcer, Cheu, Joseph, Mao]

- The protocol we saw for counting bits has nice feature: It is resilient to dropouts
- Specifically, privacy is still preserved if a constant fraction of the users (say half) do not submit messages to the shuffle

Recall the privacy requirement we had before:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$ and any event $\boldsymbol{F}$ we have

$$\Pr\left[\text{shuffle}\left(R(x_1), \ldots, R(x_n)\right) \in F\right] \leq e^{\varepsilon} \cdot \Pr\left[\text{shuffle}\left(R(x_1'), \ldots, R(x_n')\right) \in F\right] + \delta$$

The Robust Shuffle Model

[Balcer, Cheu, Joseph, Mao]

- The protocol we saw for counting bits has nice feature: It is resilient to dropouts
- Specifically, privacy is still preserved if a constant fraction of the users (say half) do not submit messages to the shuffle

Recall the privacy requirement we had before:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$ and any event $\boldsymbol{F}$ we have

$$\Pr\left[\text{shuffle}\left(R(x_1), \dots, R(x_n)\right) \in F\right] \le e^{\varepsilon} \cdot \Pr\left[\text{shuffle}\left(R(x_1'), \dots, R(x_n')\right) \in F\right] + \delta$$

A modified privacy definition – the robust shuffle model:

For any neighboring datasets $\vec{\boldsymbol{x}}, \vec{\boldsymbol{x}}'$, event $\boldsymbol{F}$, and set of induces $\{\boldsymbol{j_1}, \boldsymbol{j_2}, \dots, \boldsymbol{j_w}\}$ of size $\boldsymbol{w} \ge \frac{\boldsymbol{n}}{\boldsymbol{2}}$

$$\Pr\left[\text{shuffle}\left(R(x_{j_1}), \dots, R(x_{j_w})\right) \in F\right] \le e^{\varepsilon} \cdot \Pr\left[\text{shuffle}\left(R\left(x_{j_1}'\right), \dots, R\left(x_{j_w}'\right)\right) \in F\right] + \delta$$

The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

Negative result for the shuffle model

[Balcer, Cheu, Joseph, Mao]

The XOR-sum problem:

- The input of every user i is a pair $(j_i, b_i) \in \{1,2,\dots,n\} \times \{0,1\}$
- The goal: estimate

$$\sum_{j=1}^{n} \bigoplus_{i:j_i=j} b_i$$

Example: if the inputs are $(1,1), (1,0), (3,0), (1,1), (3,0), (2,1), (4,1)$
Then the goal is to estimate $(1 \oplus 0 \oplus 1) + (1) + (0 \oplus 0) + (1) = 2$

Negative result for the shuffle model

[Balcer, Cheu, Joseph, Mao]

The XOR-sum problem:

- The input of every user i is a pair $(j_i, b_i) \in \{1,2,\ldots,n\} \times \{0,1\}$
- The goal: estimate

$$\sum_{j=1}^{n} \bigoplus_{i:j_i=j} b_i$$

Example: if the inputs are $(1,1),(1,0),(3,0),(1,1),(3,0),(2,1),(4,1)$
Then the goal is to estimate $(1 \oplus 0 \oplus 1) + (1) + (0 \oplus 0) + (1) = 2$

Observe: Removing one person's data changes this quantity by ± 1 and so this can be estimated with error $\approx \frac{1}{\varepsilon}$ in the centralized model

Informal theorem: Any robust-shuffle protocol for this problem must have error $\Omega\left(\sqrt{n}\right)$

Negative result for the shuffle model

[Balcer, Cheu, Joseph, Mao]

Proof idea:

- Suppose there is a robust-shuffle algorithm $(R, \mathcal{A})$, where R is the randomizer applied by the users and $\mathcal{A}$ is the post-processing algorithm after the shuffle (on the server's side)
- Let $X = \left(x_1, \dots, x_{n/2}\right) \in \{0,1\}^{n/2}$ be an input dataset
- We can use $(R, \mathcal{A})$ to answer **many** "Hamming distance queries" w.r.t. X of the form:

Given $\boldsymbol{Y} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}/\mathbf{2}}$ approximate $(\boldsymbol{x}_{\mathbf{1}} \oplus \boldsymbol{y}_{\mathbf{1}}) + \cdots + \left(\boldsymbol{x}_{\boldsymbol{n}/\mathbf{2}} \oplus \boldsymbol{y}_{\boldsymbol{n}/\mathbf{2}}\right)$

Negative result for the shuffle model

[Balcer, Cheu, Joseph, Mao]

Proof idea:

- Suppose there is a robust-shuffle algorithm $(R, \mathcal{A})$, where R is the randomizer applied by the users and $\mathcal{A}$ is the post-processing algorithm after the shuffle (on the server's side)
- Let $X = \left(x_1, \ldots, x_{n/2}\right) \in \{0,1\}^{n/2}$ be an input dataset
- We can use $(R, \mathcal{A})$ to answer **many** "Hamming distance queries" w.r.t. X of the form:

Given $\boldsymbol{Y} \in \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}/\mathbf{2}}$ approximate $(\boldsymbol{x}_{\mathbf{1}} \oplus \boldsymbol{y}_{\mathbf{1}}) + \cdots + \left(\boldsymbol{x}_{\boldsymbol{n}/\mathbf{2}} \oplus \boldsymbol{y}_{\boldsymbol{n}/\mathbf{2}}\right)$

1) Output $\mathrm{G} \leftarrow \mathrm{Shuffle}\left(R(1, x_1), \ldots, R\left(\frac{n}{2}, x_{n/2}\right)\right)$ % by assumption, G is safe for publication
2) Given a query $Y = \left(y_1, \ldots, y_{n/2}\right) \in \{0,1\}^{n/2}$ respond with

$$\mathcal{A}\left(\mathrm{Shuffle}\left(G, R(1, y_1), \ldots, R\left(n/2, y_{n/2}\right)\right)\right)$$

- Step 2 is just a post-processing of the output of step 1 and hence the algorithm remains private regardless of how many queries we answer in Step 2
- Answering "too many" queries with "too much" accuracy is impossible…

The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction

Interaction in the shuffle model

- The negative result for XOR-sum strongly relied on the protocol being non-interactive
- This allowed us to “pause” the computation mid-way and continue arbitrarily
- Does not work with interactive protocols

Interaction in the shuffle model

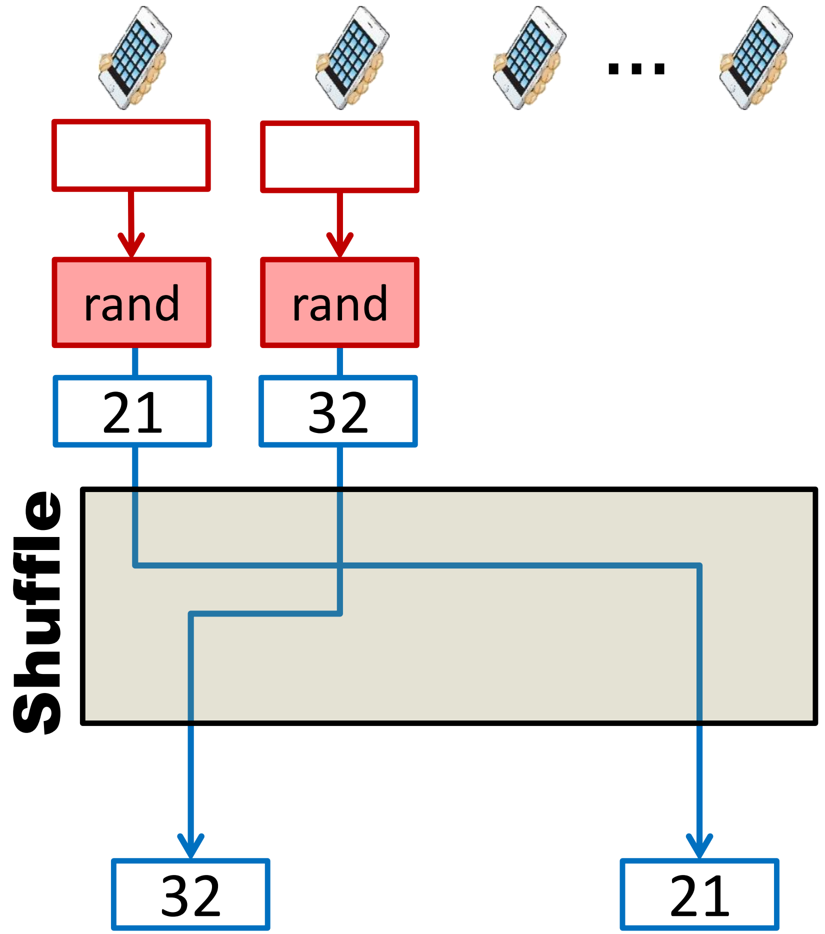
- The negative result for XOR-sum strongly relied on the protocol being non-interactive
- This allowed us to “pause” the computation mid-way and continue arbitrarily
- Does not work with interactive protocols

Informal theorem: Every randomized functionality can be computed in the shuffle model with merely two rounds

Interaction in the shuffle model

- The negative result for XOR-sum strongly relied on the protocol being non-interactive
- This allowed us to “pause” the computation mid-way and continue arbitrarily
- Does not work with interactive protocols

Informal theorem: Every randomized functionality can be computed in the shuffle model with merely two rounds



Key Exchange: Definition

- User i and user j send messages to the shuffle
- All users see the shuffled messages
- User i and user j agree on a key
- All other users together get no information on the key

Interaction in the shuffle model

Agreeing on one bit

[IKOS'06]

- User $\boldsymbol{i}$ chooses a random bit $\boldsymbol{a}$, sends it to the shuffle
- User $\boldsymbol{j}$ chooses a random bit $\boldsymbol{b}$, sends it to the shuffle
- If $\boldsymbol{a} \neq \boldsymbol{b}$ the common key is $\boldsymbol{a}$, otherwise protocol fails

Interaction in the shuffle model

Agreeing on one bit

[IKOS'06]

- User $\boldsymbol{i}$ chooses a random bit $\boldsymbol{a}$, sends it to the shuffle
- User $\boldsymbol{j}$ chooses a random bit $\boldsymbol{b}$, sends it to the shuffle
- If $\boldsymbol{a} \neq \boldsymbol{b}$ the common key is $\boldsymbol{a}$, otherwise protocol fails

- User $\boldsymbol{i}$ knows $\boldsymbol{a}$. User $\boldsymbol{j}$ that knows $\boldsymbol{b}$ and sees output of shuffle learns $\boldsymbol{a}$
- All users see $\boldsymbol{a}, \boldsymbol{b}$ with prob. ½ and $\boldsymbol{b}, \boldsymbol{a}$ with prob. ½
  - All other users get no info. on $\boldsymbol{a}$
- Users $\boldsymbol{i}, \boldsymbol{j}$ agree on a key with prob. ½

Interaction in the shuffle model

Agreeing on one bit

[IKOS'06]

- User $\boldsymbol{i}$ chooses a random bit $\boldsymbol{a}$, sends it to the shuffle
- User $\boldsymbol{j}$ chooses a random bit $\boldsymbol{b}$, sends it to the shuffle
- If $\boldsymbol{a} \neq \boldsymbol{b}$ the common key is $\boldsymbol{a}$, otherwise protocol fails

- User $\boldsymbol{i}$ knows $\boldsymbol{a}$. User $\boldsymbol{j}$ that knows $\boldsymbol{b}$ and sees output of shuffle learns $\boldsymbol{a}$
- All users see $\boldsymbol{a}, \boldsymbol{b}$ with prob. ½ and $\boldsymbol{b}, \boldsymbol{a}$ with prob. ½
  - All other users get no info. on $\boldsymbol{a}$
- Users $\boldsymbol{i}, \boldsymbol{j}$ agree on a key with prob. ½

Agreeing on k bits

- User $\boldsymbol{i}$ chooses $\boldsymbol{3k}$ random bits $\boldsymbol{a_1}, \dots, \boldsymbol{a_{3k}}$, sends $(\boldsymbol{1}, \boldsymbol{a_1}), \dots, (\boldsymbol{3k}, \boldsymbol{a_{3k}})$ to the shuffle
- User $\boldsymbol{j}$ chooses $\boldsymbol{3k}$ random bits $\boldsymbol{b_1}, \dots, \boldsymbol{b_{3k}}$, sends $(\boldsymbol{1}, \boldsymbol{b_1}), \dots, (\boldsymbol{3k}, \boldsymbol{b_{3k}})$ to the shuffle
- Let $\boldsymbol{I}$ be the first $\boldsymbol{k}$ indices s.t. $\boldsymbol{a_i} \neq \boldsymbol{b_i}$; the common key is $(\boldsymbol{a_i})_{\boldsymbol{i \in I}}$

- Users $\boldsymbol{i}, \boldsymbol{j}$ agree on a secret $\boldsymbol{k}$-bit key with prob. $\boldsymbol{1 - 2^{-O(k)}}$

Interaction in the shuffle model

Agreeing on one bit

[IKOS'06]

- User $\boldsymbol{i}$ chooses a random bit $\boldsymbol{a}$, sends it to the shuffle
- User $\boldsymbol{j}$ chooses a random bit $\boldsymbol{b}$, sends it to the shuffle
- If $\boldsymbol{a} \neq \boldsymbol{b}$ the common key is $\boldsymbol{a}$, otherwise protocol fails

Theorem: With the addition of one round (for setting private channels) general MPC can be implemented in the shuffle model

* additional round can be avoided in some cases

Agreeing on k bits

- User $\boldsymbol{i}$ chooses $\boldsymbol{3k}$ random bits $\boldsymbol{a_1}, \dots, \boldsymbol{a_{3k}}$, sends $(\boldsymbol{1}, \boldsymbol{a_1}), \dots, (\boldsymbol{3k}, \boldsymbol{a_{3k}})$ to the shuffle
- User $\boldsymbol{j}$ chooses $\boldsymbol{3k}$ random bits $\boldsymbol{b_1}, \dots, \boldsymbol{b_{3k}}$, sends $(\boldsymbol{1}, \boldsymbol{b_1}), \dots, (\boldsymbol{3k}, \boldsymbol{b_{3k}})$ to the shuffle
- Let $\boldsymbol{I}$ be the first $\boldsymbol{k}$ indices s.t. $\boldsymbol{a_i} \neq \boldsymbol{b_i}$; the common key is $(\boldsymbol{a_i})_{\boldsymbol{i \in I}}$

- Users $\boldsymbol{i}, \boldsymbol{j}$ agree on a secret $\boldsymbol{k}$-bit key with prob. $\boldsymbol{1 - 2^{-O(k)}}$

The Shuffle Model of Differential Privacy

Today's Outline

1. Secure Multiparty Computation (MPC)
2. What is the shuffle model
3. Counting bits
4. Robustness in the shuffle model
5. Negative result for the shuffle model
6. Interaction