Network Shuffling

Privacy amplification via Random Walks

SIGMOD 2022

Seng Pei Liew

with Tsubasa Takahashi , Shun Takagi*, Fumiyuki Kato*, Yang Cao*, Masatoshi Yoshikawa*

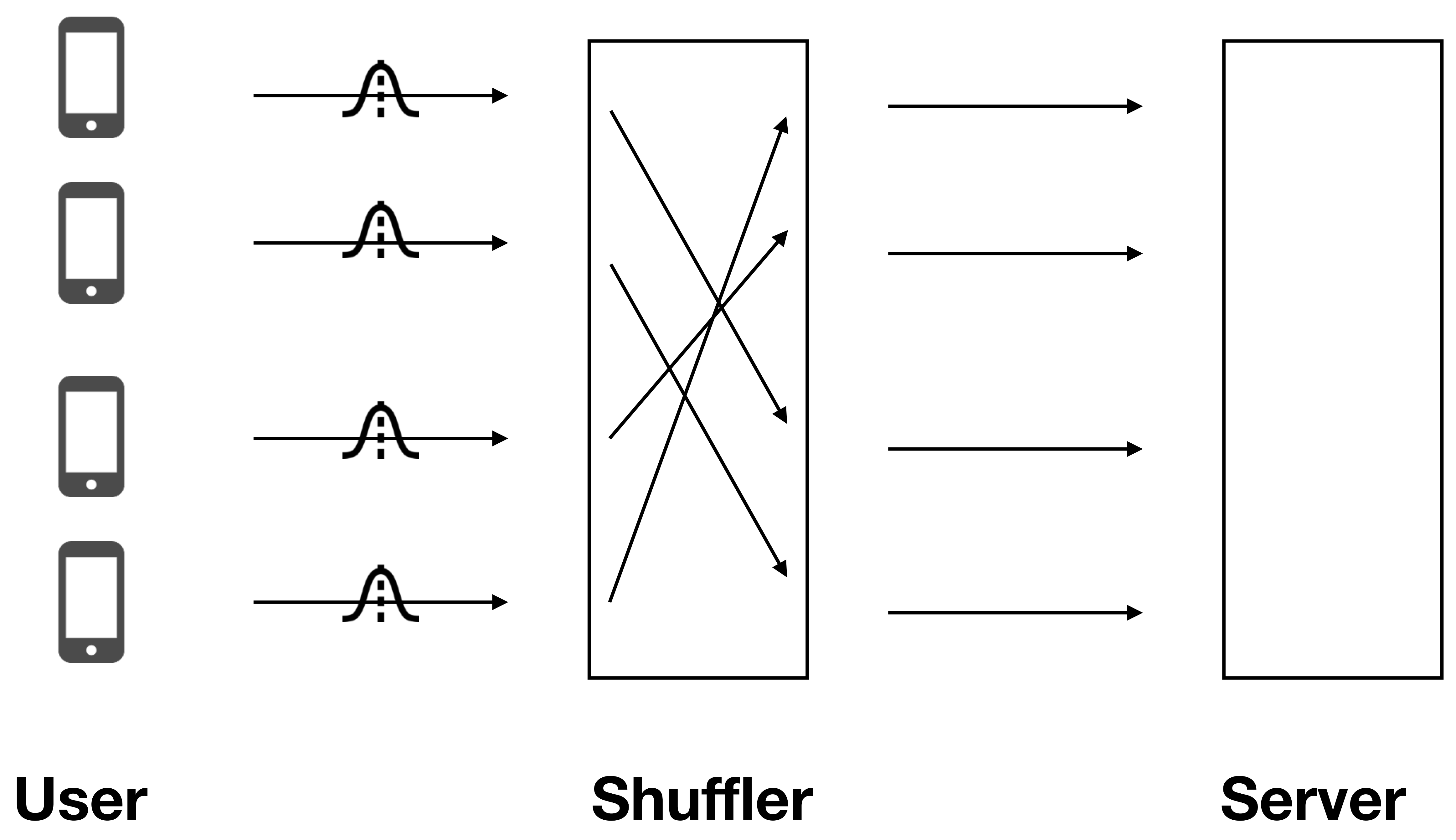
LINE Corporation, Japan

** Kyoto University, Japan*

LINE

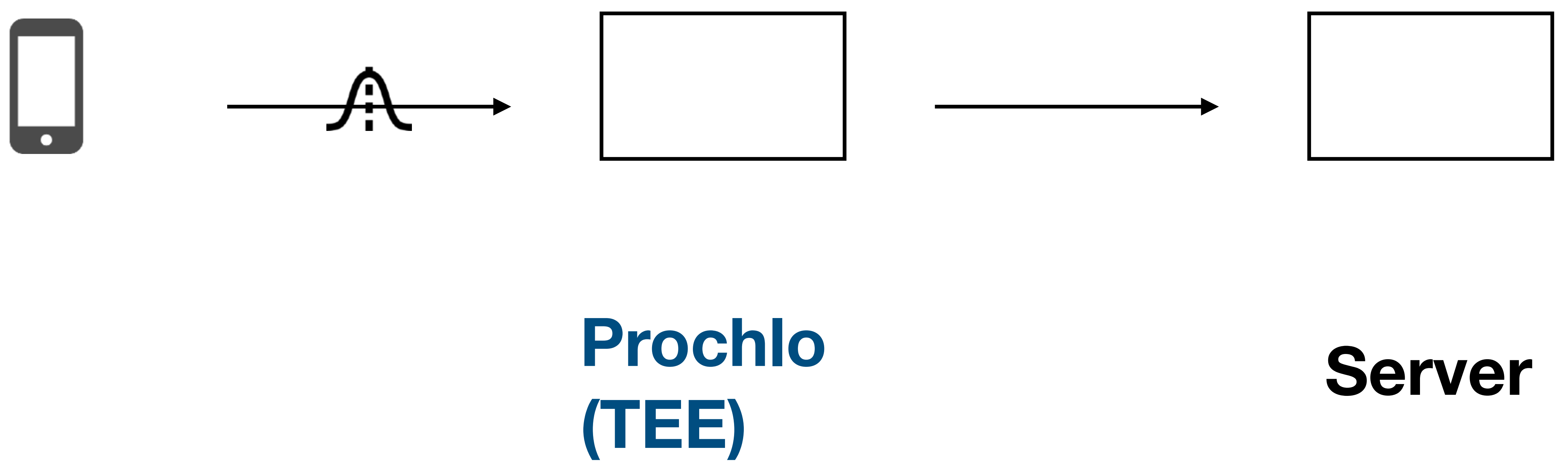
How to anonymize data to enhance differential privacy?

- User wants to send (randomized) data to the server anonymously (**Shuffle model**)
- Anonymization is typically assumed to be performed with a centralized **shuffler**



It is shown that anonymization leads to privacy amplification in terms of differential privacy

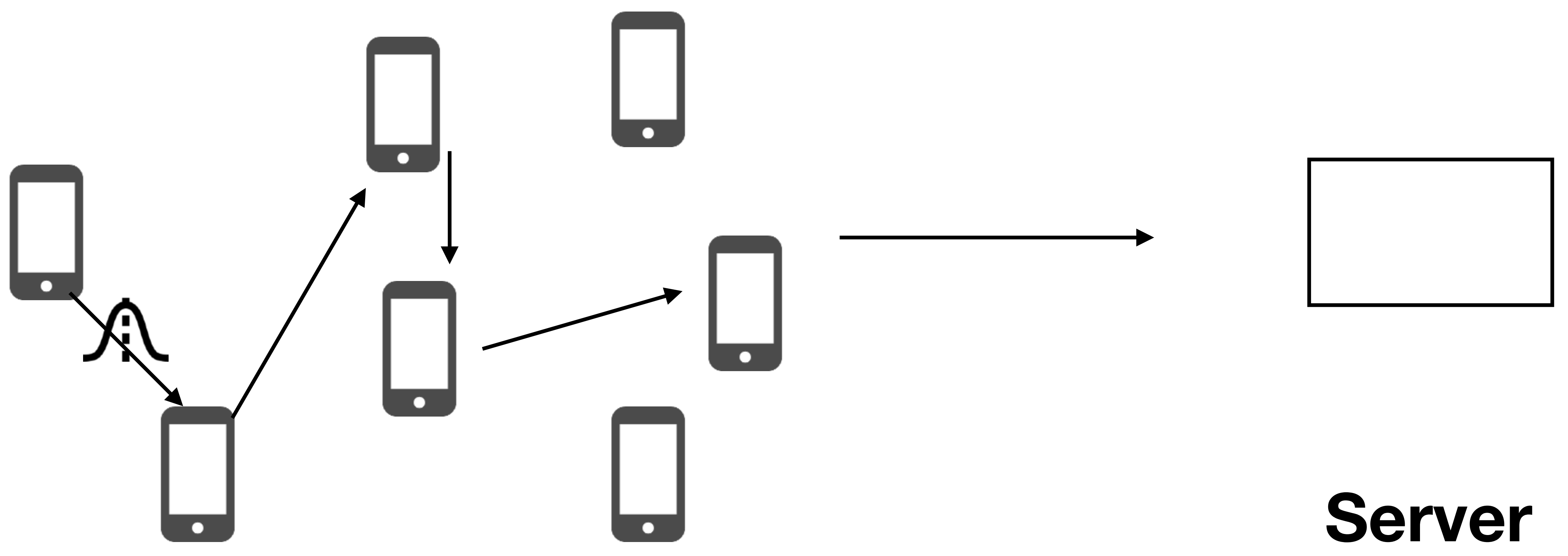
Trusted shuffler implementation



- Vulnerable to side-channel attacks
- single-point failure

[BEM+17]

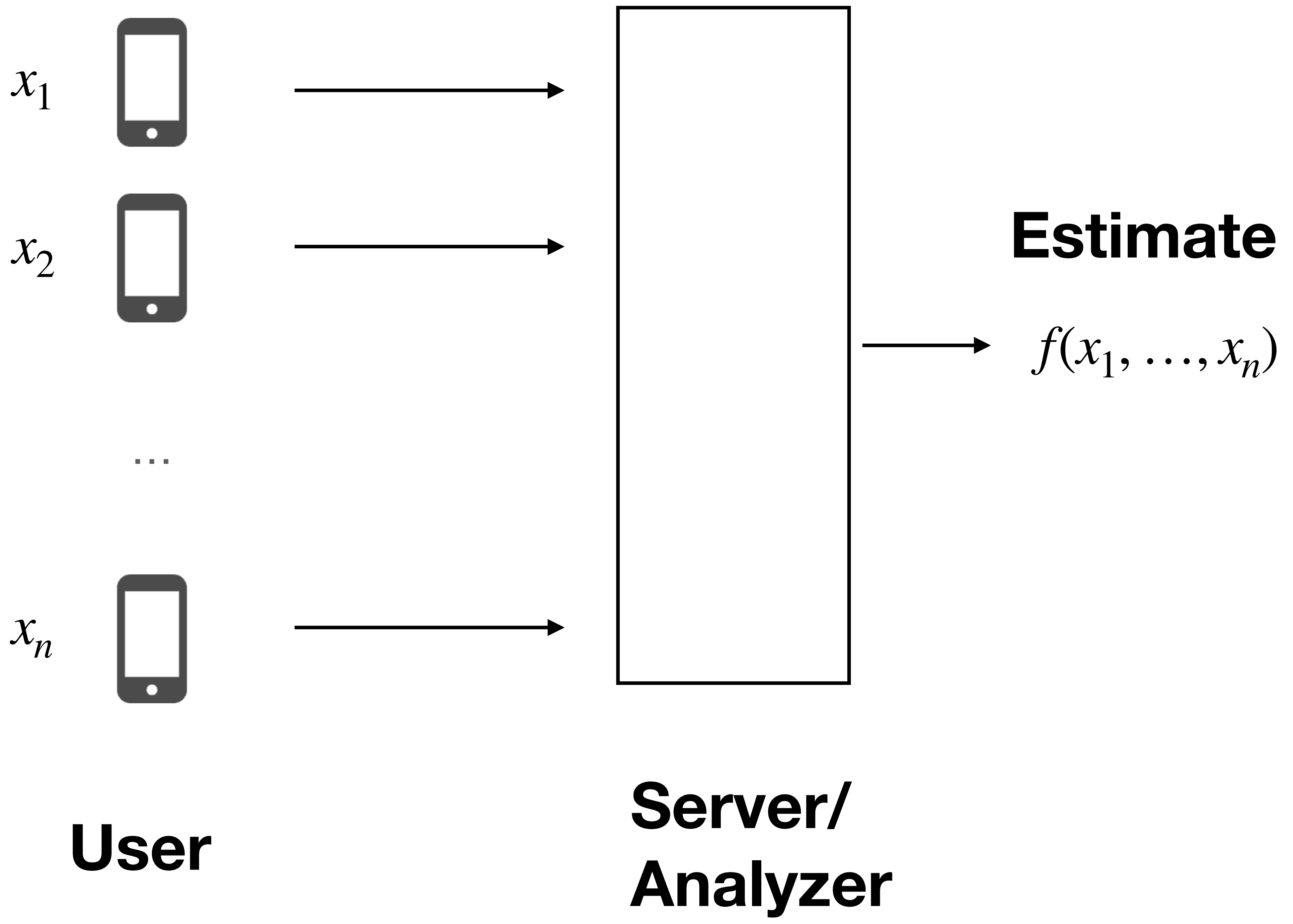
Network shuffling (our proposal)



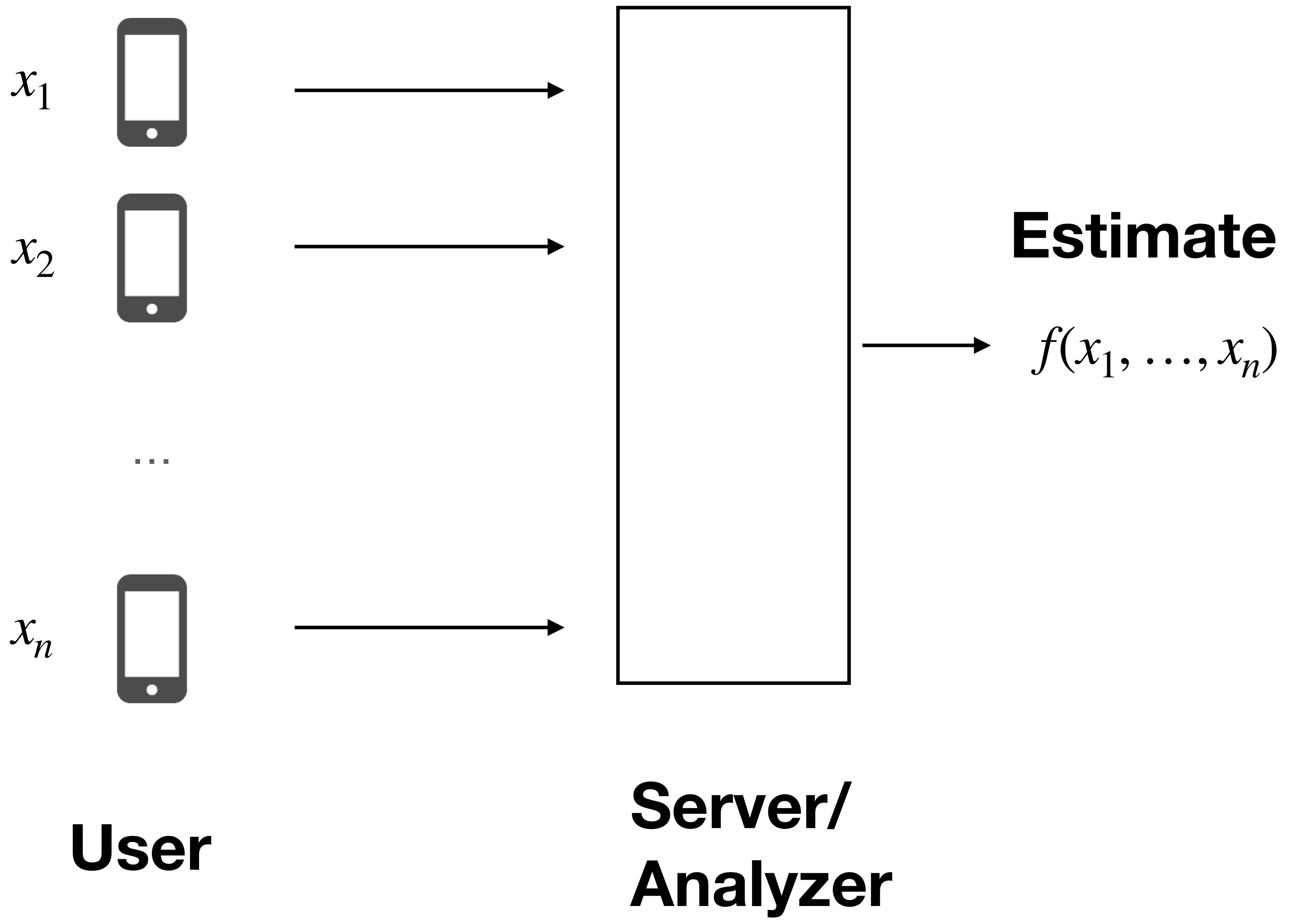
- No centralized entity required

We give analytical results showing that privacy amplification is achievable under this decentralized setting

Distributed Analytics



Differential Privacy

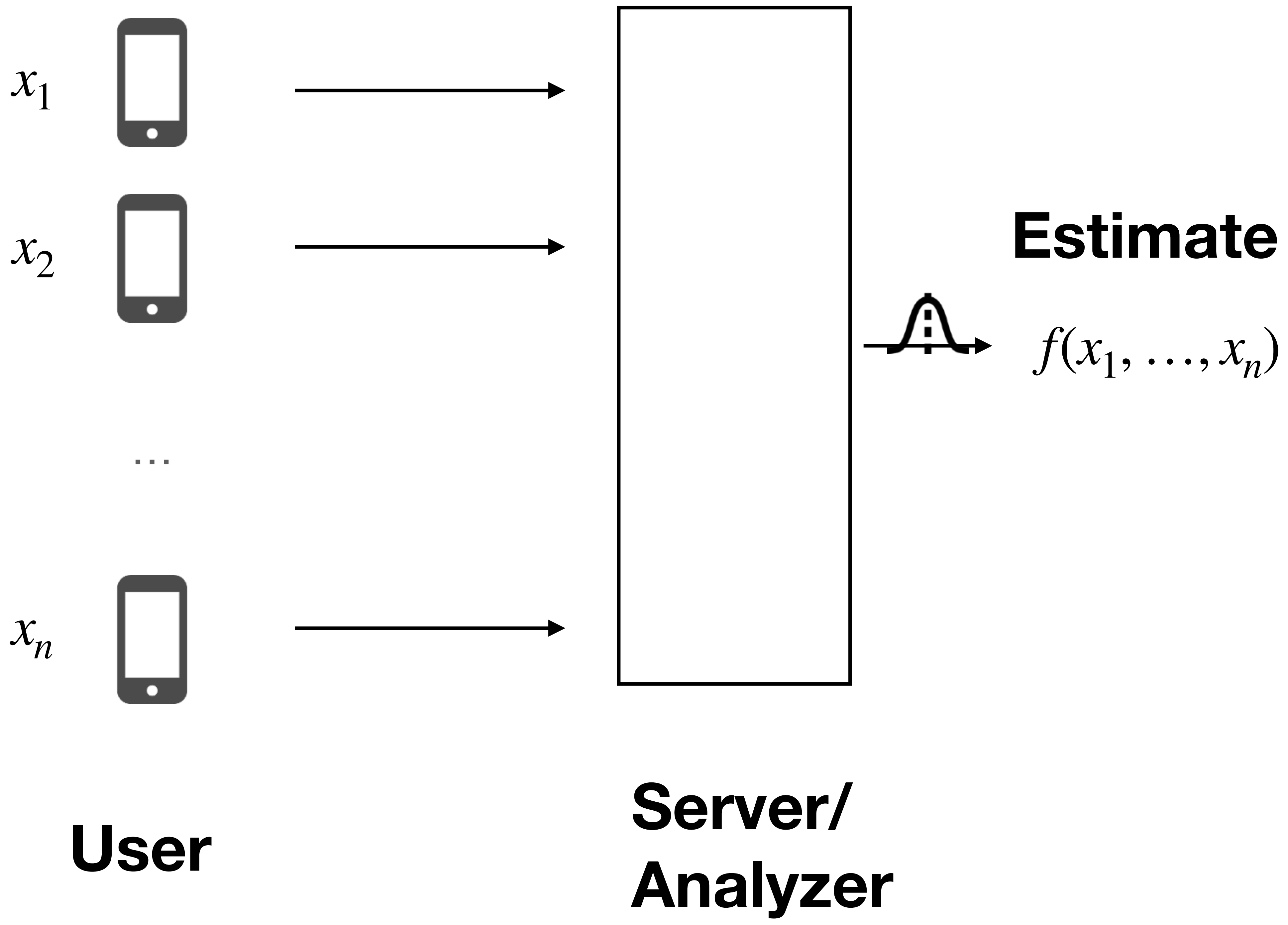


(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$

Differential Privacy (central)



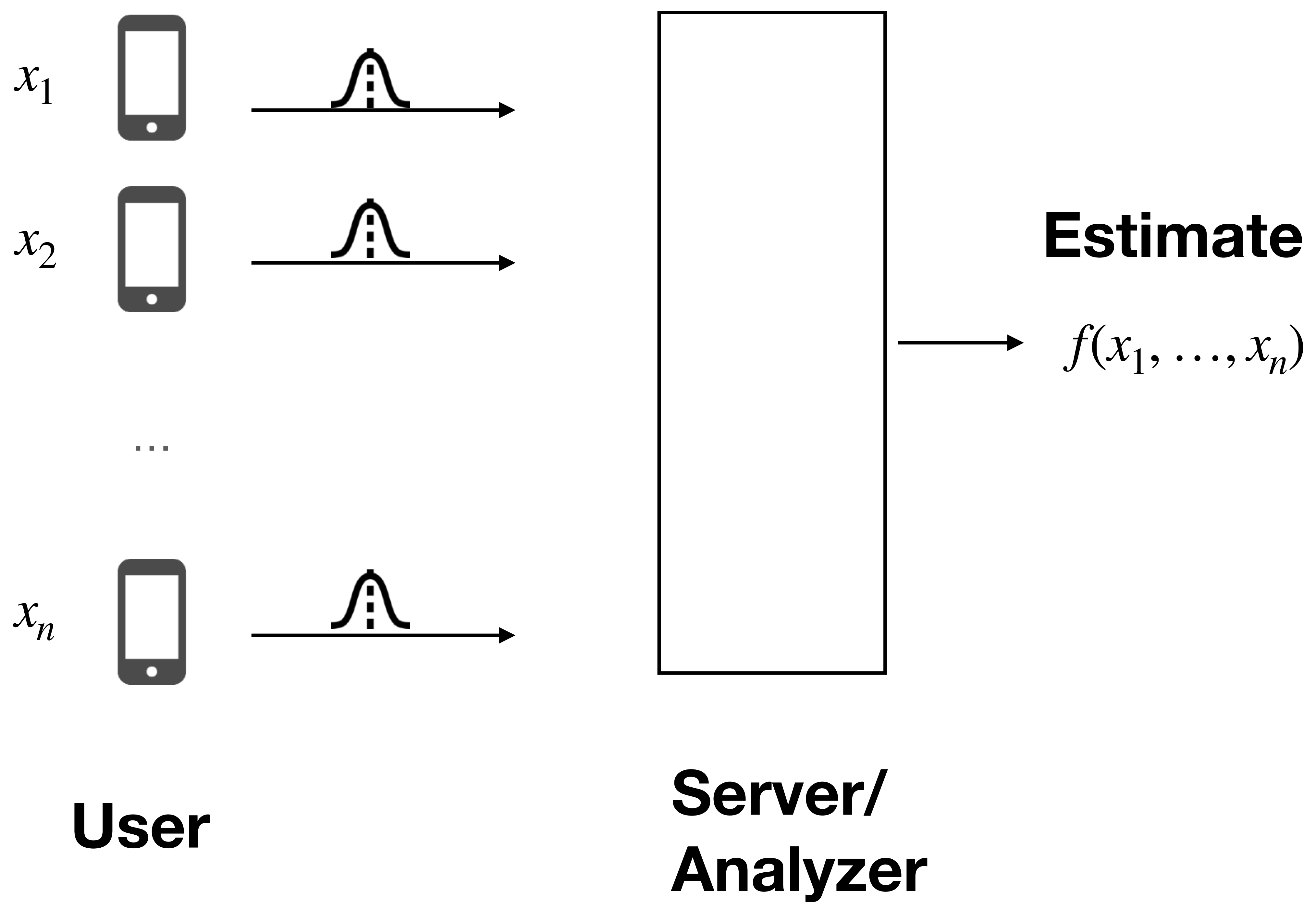
(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$$\Pr[M(D) = x] \leq e^{\epsilon} \Pr[M(D') = x] + \delta$$

- Pro: Utility is high (comparably small amount of noise is required to maintain indistinguishability)
- Con: One must trust the server (for not leaking privacy)

Differential Privacy (local)



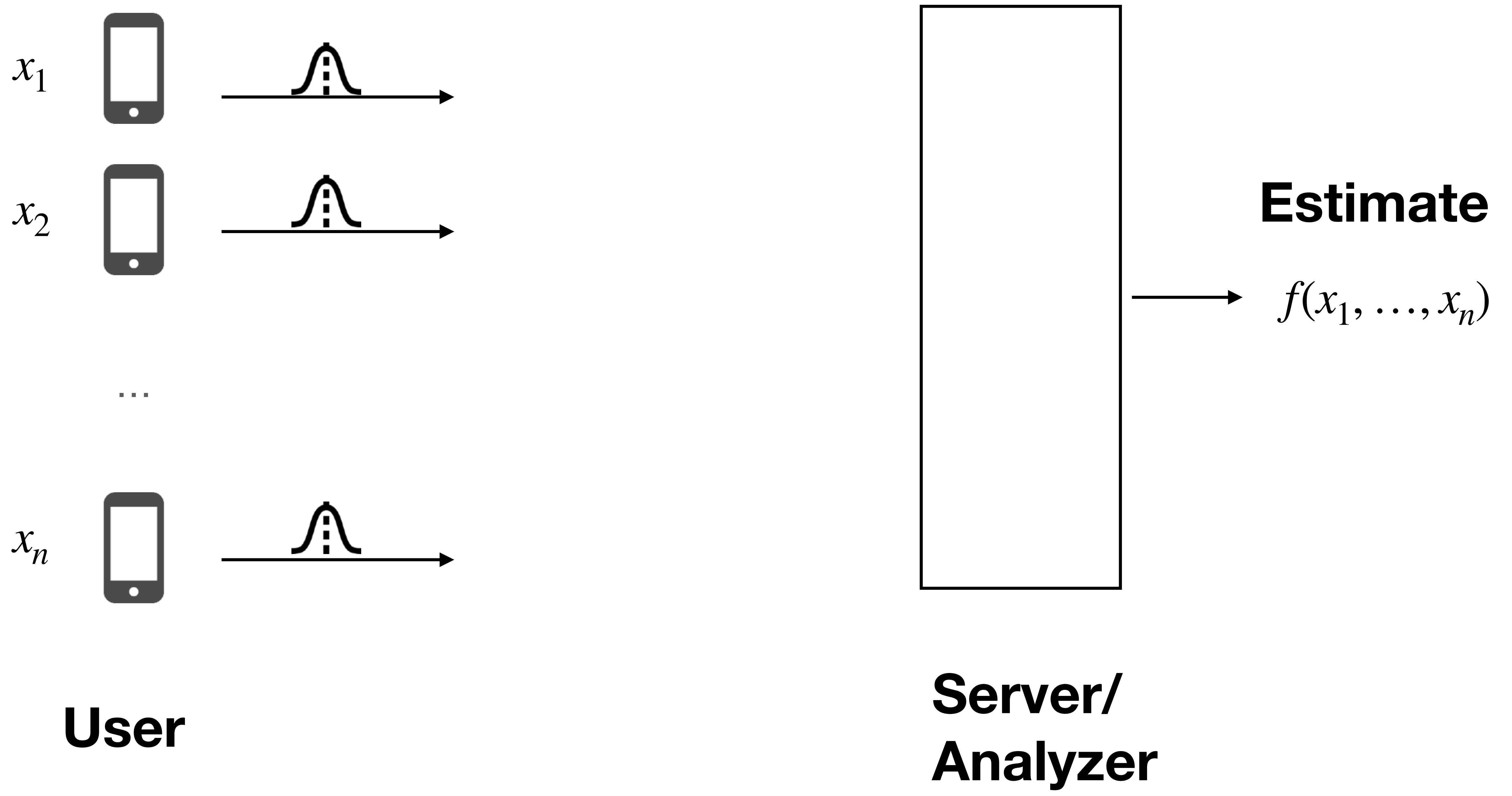
(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$$

- Pro: No trust assumption on aggregator is assumed
- Con: Low utility (Noise required to maintain indistinguishability is relatively high)

Differential Privacy (shuffle)

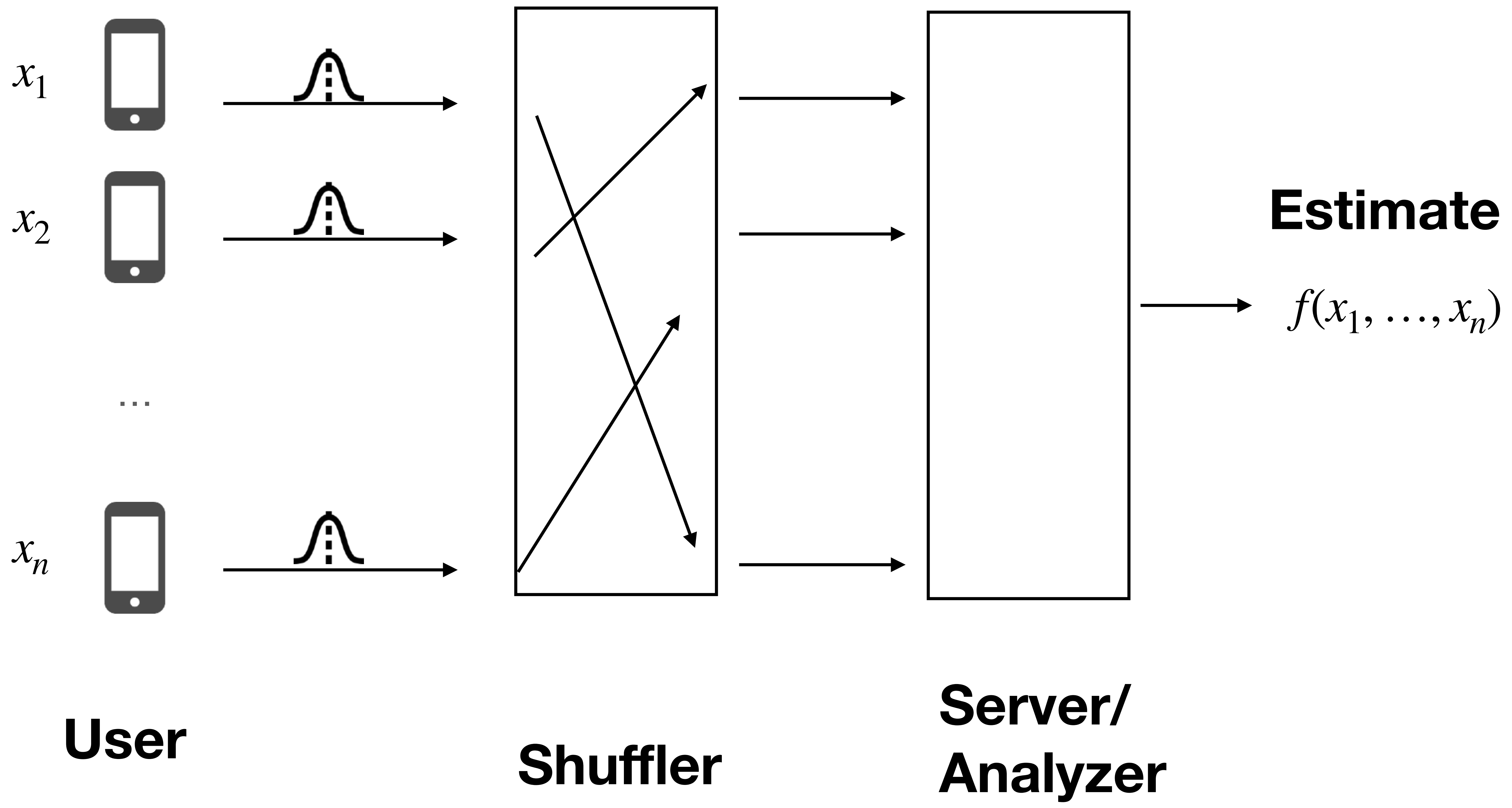


(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$

Differential Privacy (shuffle)



(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$

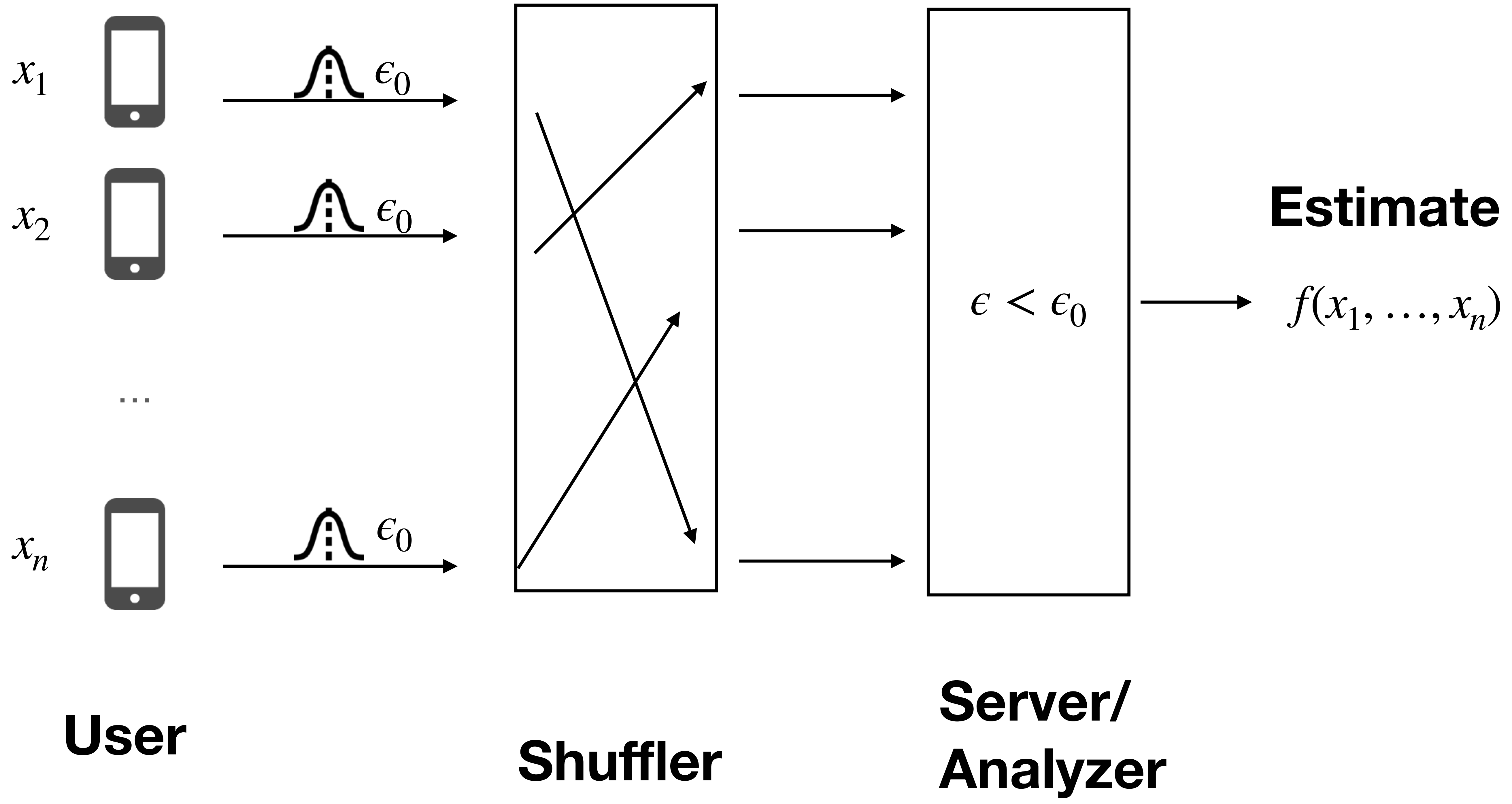
Encode, Shuffle,
Analyze

Anonymization

Shuffle/shuffled model

[BEM+17]
[CSU+19]
[EFM+19]

Differential Privacy (shuffle)



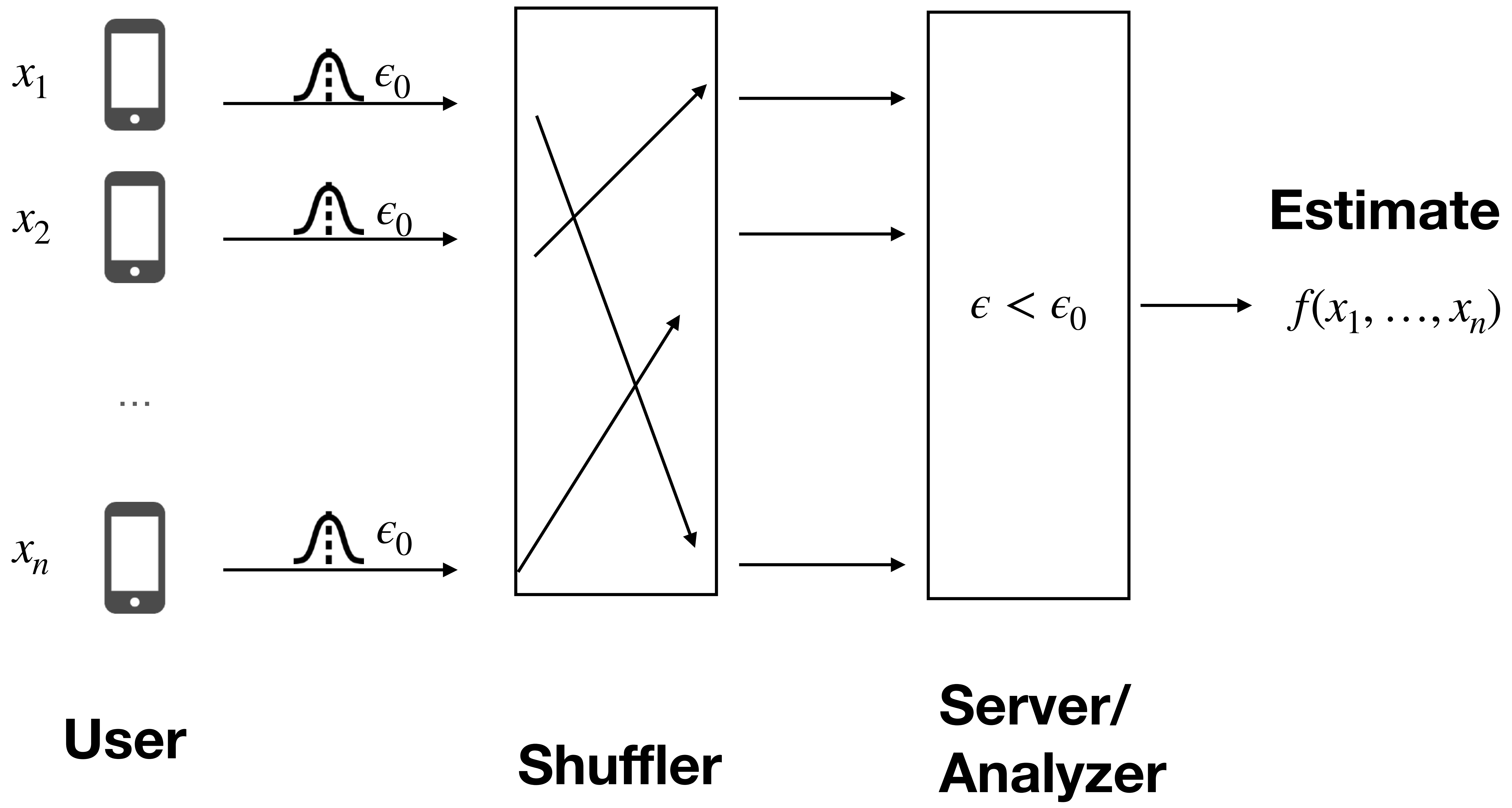
(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$

- The shuffler removes any identifier (identifying the user sending the data). Also known as *uniform shuffling*.
- **Privacy amplification** is said to occur when $\epsilon < \epsilon_0$ (ϵ being the overall, central DP, ϵ_0 being the individual LDP)

Differential Privacy (shuffle)



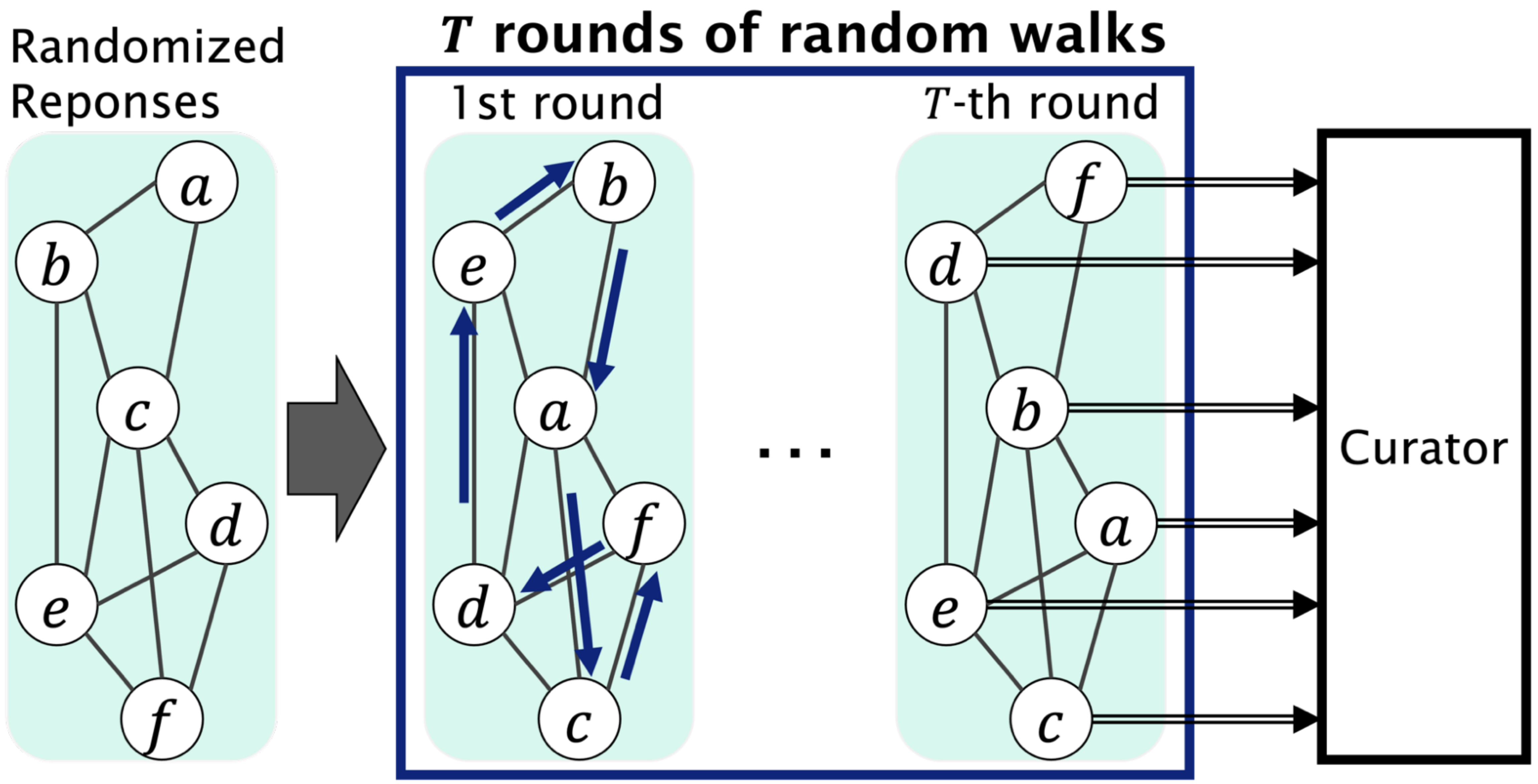
(ϵ, δ)-Differential Privacy

- "An algorithm is differential private if changing a single record does not alter its output distribution by much." [DN03, DMNS06]

$$\Pr[M(D) = x] \leq e^{\epsilon}\Pr[M(D') = x] + \delta$$

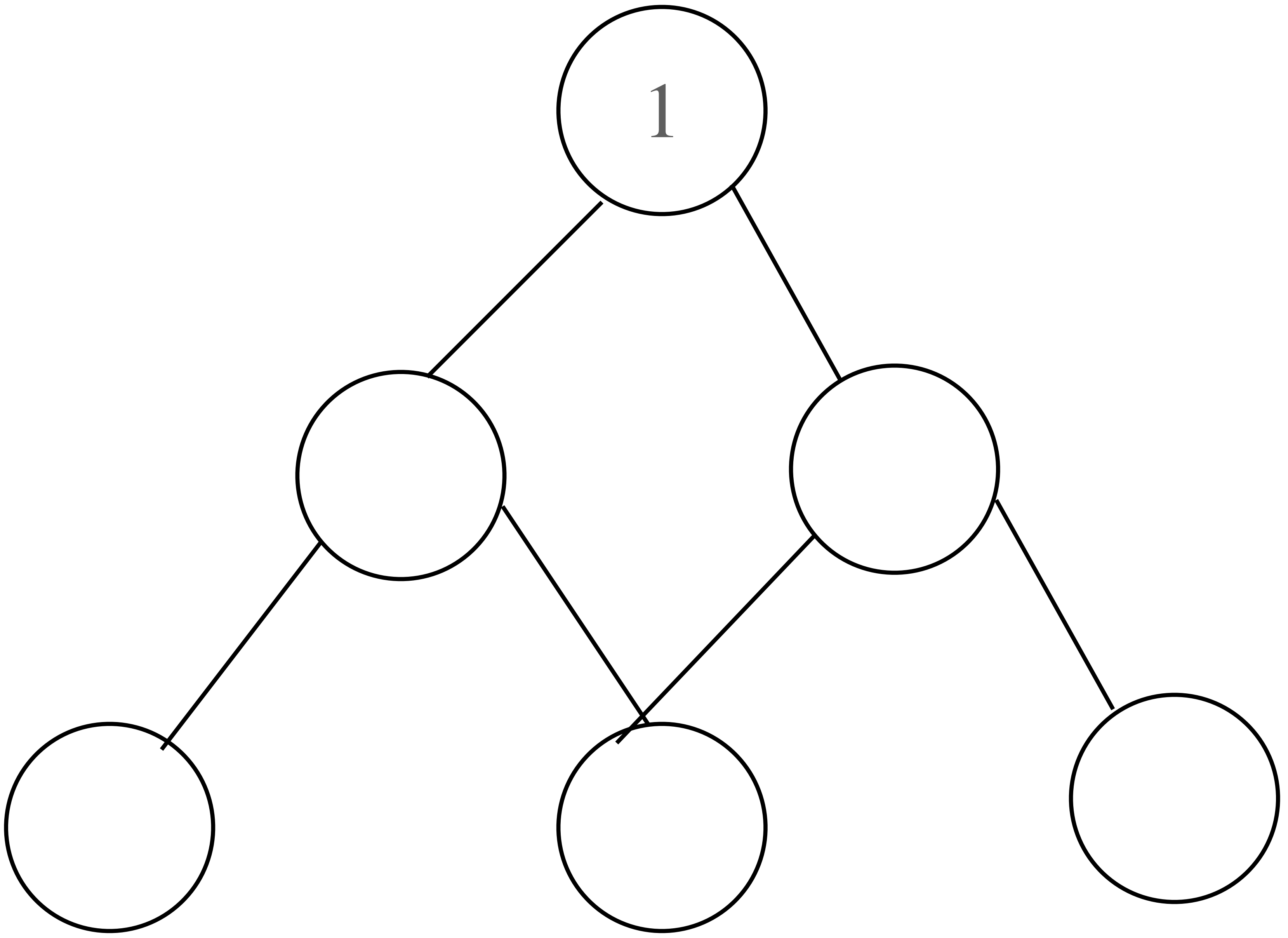
- Pro: Better utility than the local DP
- Con: One must trust the shuffler instead (e.g., using trusted hardware)

Proposal (network shuffling)



- We would like to achieve the same shuffling effect ***without*** using a centralized shuffler.
- The main idea is to **exchange the user output within each other on a network** before sending the (exchanged) data to the server
- The server receive messages from the users **without knowing the origin of the messages**, thus achieving anonymization.
- Our proposal is motivated by messaging apps (LINE, Facebook Messenger) where users exchange messages on a social network

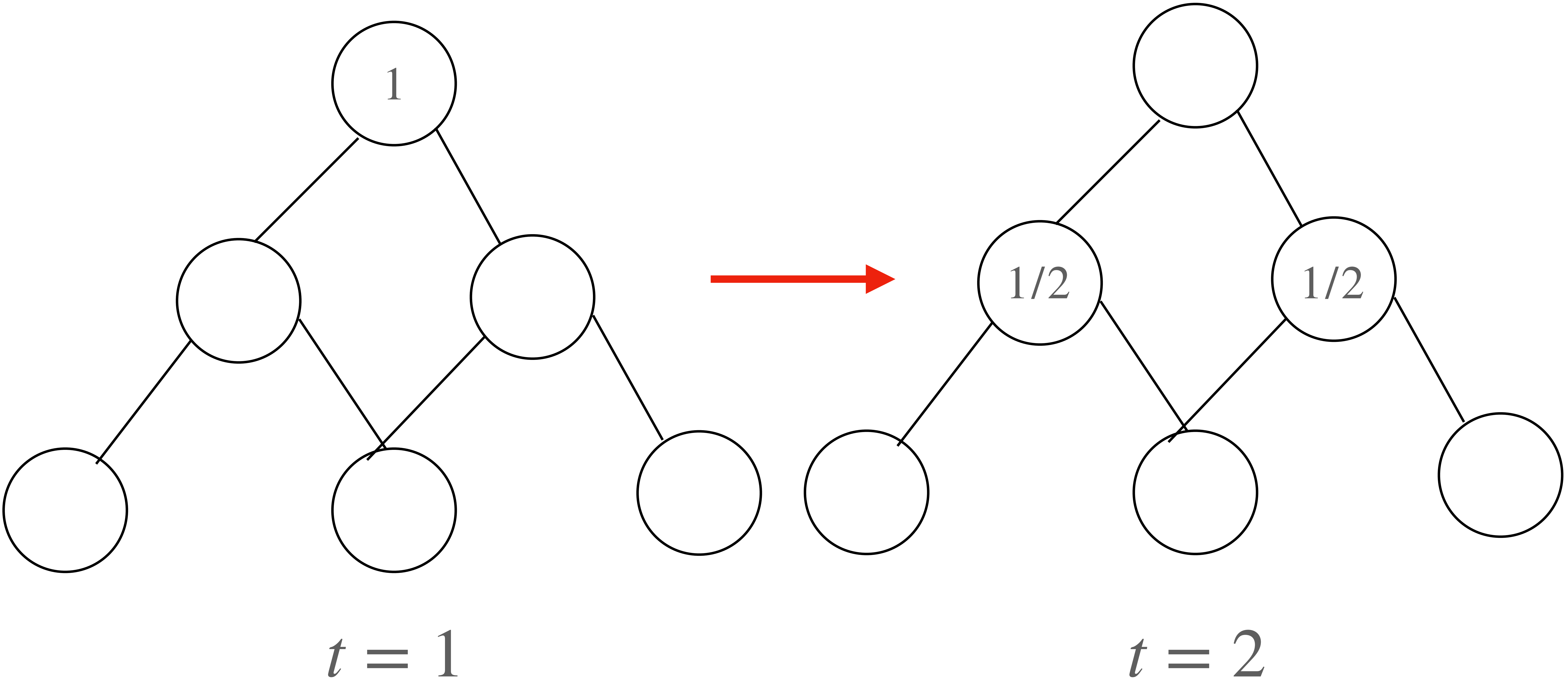
Modeling network shuffling as a random walk on graphs



$t = 1$

- Assume a fixed communication network/graph, and that all users exchange messages randomly and uniformly with neighbors.
- This corresponds to the well-studied topic of random walk on graphs.

Modeling network shuffling as a random walk on graphs

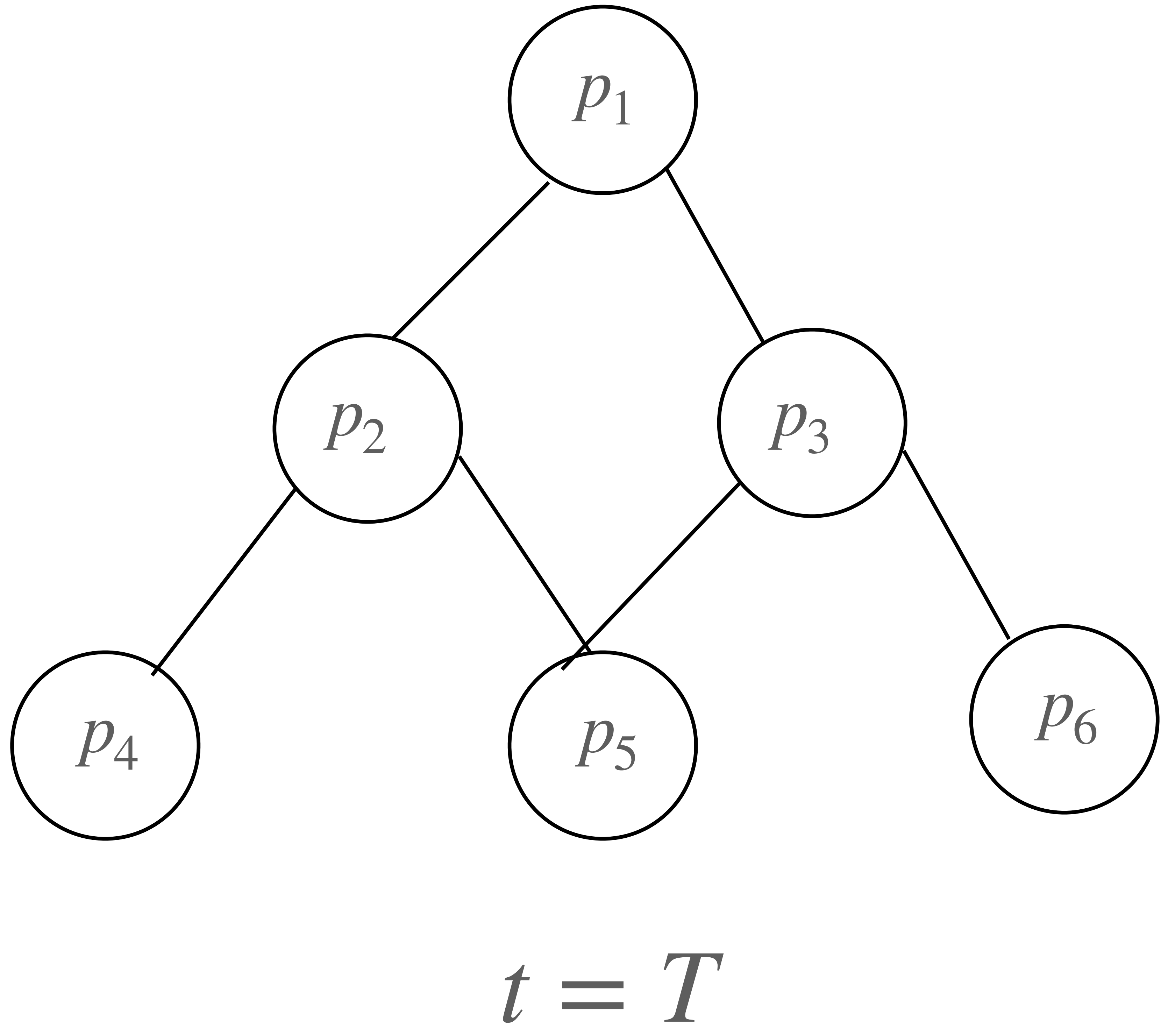


- Assume a fixed communication network/graph, and that all users exchange messages randomly and uniformly with neighbors.
- This corresponds to the well-studied topic of random walk on graphs.

Modeling network shuffling as a random walk on graphs

- Assume a fixed communication network/graph, and that all users exchange messages randomly and uniformly with neighbors.
- This corresponds to the well-studied topic of random walk on graphs.

The adversary view



- The privacy parameters are calculated based on the adversary (server) knowledge of the probability of a certain node receiving the message of a target user given $t = T$ (number of communication rounds).
- This is different from uniform shuffling, where the shuffling is uniform.
- Each user can also receive more than one message at one time.

Privacy amplification theorem

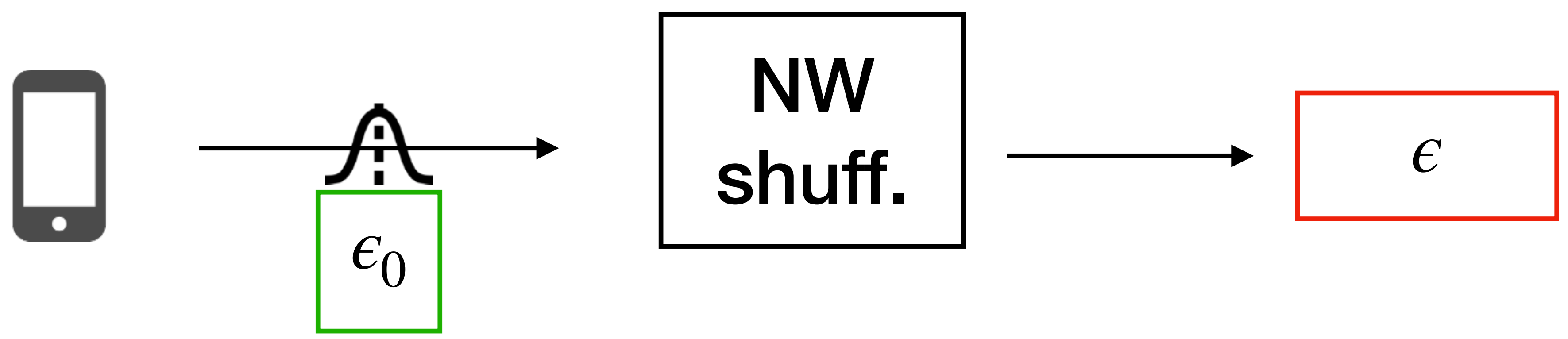
- Assume that users send all messages to the server ("all" protocol)
- The proof is based on the reduction of shuffling to swapping [EFM+19]

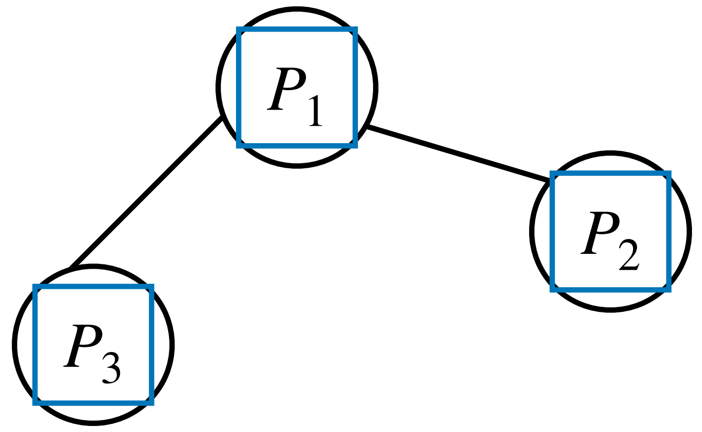
THEOREM 5.3 ("ALL" PROTOCOL, STATIONARY DISTRIBUTION). *Let $\mathcal{A}_{ldp}$ be a ε_0-local randomizer. Let $\mathcal{A}_{all} : \mathcal{D}^n \rightarrow \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the protocol as shown in Algorithm 1 sending all reports to the server. Then, $\mathcal{A}_{all}$ satisfies $(\varepsilon, \delta + \delta_2)$-DP, with*

$$\boxed{\varepsilon} = \frac{(\boxed{e^{\varepsilon_0}} - 1)^2 e^{4\varepsilon_0} \varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0} \log \frac{1}{\delta}}, \qquad (8)$$

$$\varepsilon_1 = \sqrt{\left(1 - \frac{1}{n}\right) \boxed{\sum_{i \in [n]} P_i^{G\,2}}} + \sqrt{\frac{\log(1/\delta_2)}{n}}$$

- Privacy amplification depends on network structure.
- How do we calculate this quantity?





Stationary distribution of random walk on graphs

- To calculate the probabilities, it is convenient to use the notion stationary distribution.
- Stationary distribution: a distribution π of a random walk such that for all initial distributions p_0, it converges to $\lim_{t \to \infty} = \pi$

Fact 1: A random walk on graph G converges to a stationary distribution (*ergodicity*) if and only if G is non-bipartite and connected

Fact 2: The mixing time (no. of rounds required to achieve a certain degree of homogeneity) is $\sim O(\log n)$

At any time step, we are able to show that $\sum_{i \in [n]} P_i^{G^2} \leq \sum_{i \in [n]} \pi_i^{G^2} + (1-\alpha)^{2t}$, where α is the spectral gap (roughly speaking, 1 minus the second eigenvalue of the transition matrix) to provide an upper bound (worst case) on the privacy parameter.

Privacy amplification theorem

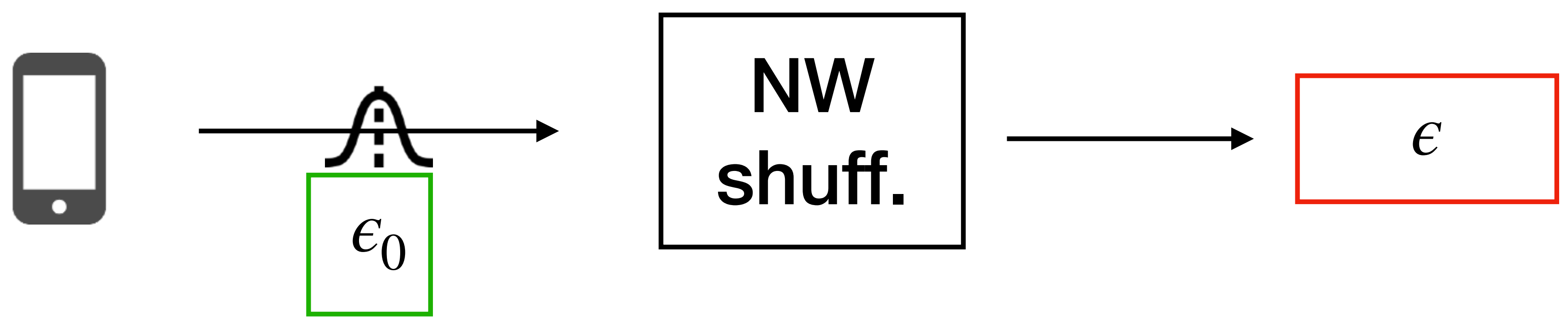
- Assume that users send all messages to the server (“all” protocol)
- The proof is based on the reduction of shuffling to swapping

Theorem 5.3 (“All” protocol, Stationary distribution). *Let $\mathcal{A}_{ldp}$ be a ε_0-local randomizer. Let $\mathcal{A}_{all} : \mathcal{D}^n \to \mathcal{S}^{(1)} \times \cdots \times \mathcal{S}^{(n)}$ be the protocol as shown in Algorithm 1 sending all reports to the server. Then, $\mathcal{A}_{all}$ satisfies $(\varepsilon, \delta + \delta_2)$-DP, with*

$$\boxed{\varepsilon} = \frac{(\boxed{e^{\varepsilon_0}} - 1)^2 e^{4\varepsilon_0} \varepsilon_1^2}{2} + \varepsilon_1 \sqrt{2(e^{\varepsilon_0} - 1)^2 e^{4\varepsilon_0} \log \frac{1}{\delta}}, \qquad (8)$$

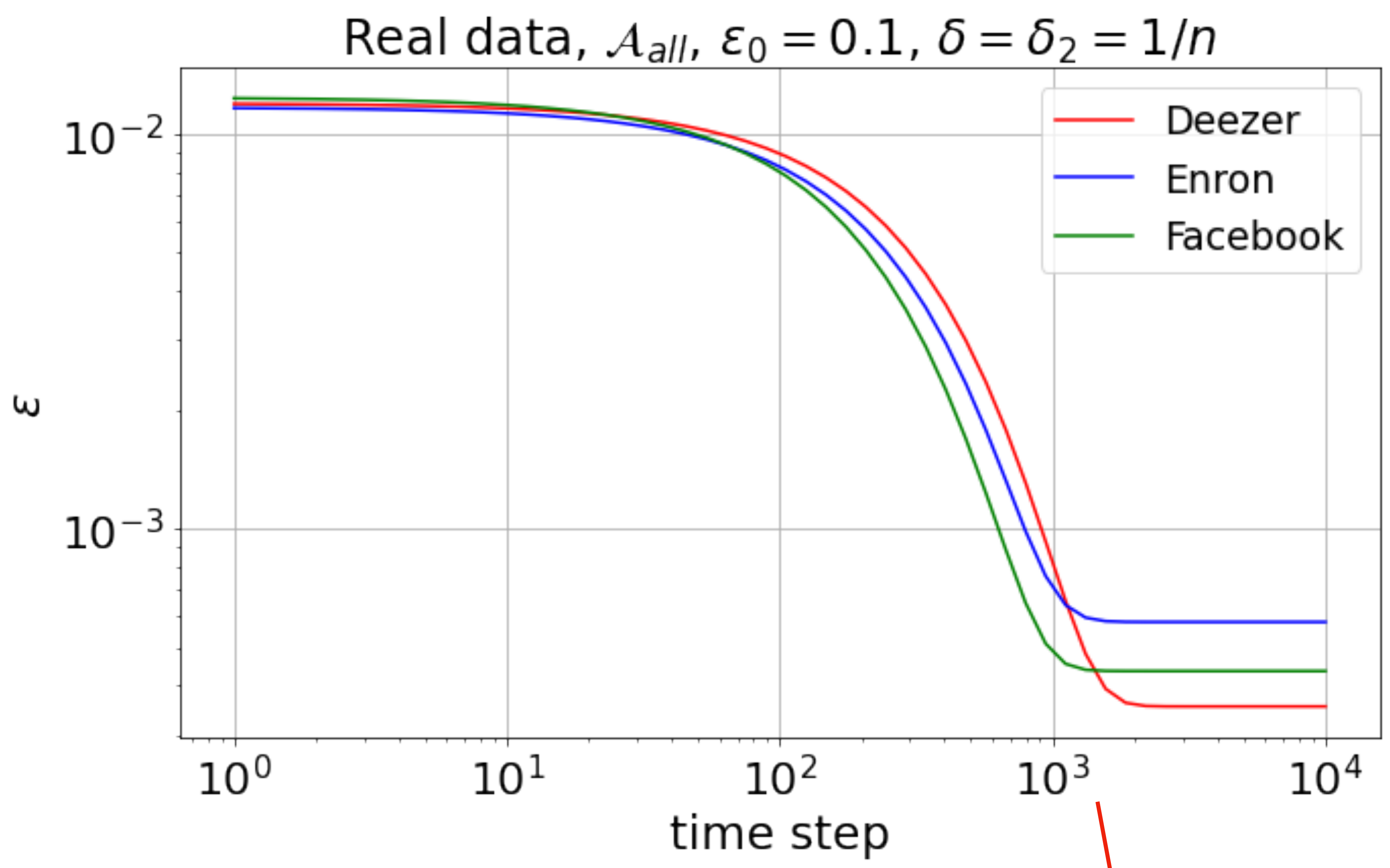
$$\varepsilon_1 = \sqrt{\left(1 - \frac{1}{n}\right) \sum_{i \in [n]} P_i^{G2}} + \sqrt{\frac{\log(1/\delta_2)}{n}},$$

$$\boxed{\sum_{i \in [n]} P_i^{G2} \leq \sum_{i \in [n]} \pi_i^{G2} + (1 - \alpha)^{2t}}$$



How the privacy guarantees change with time

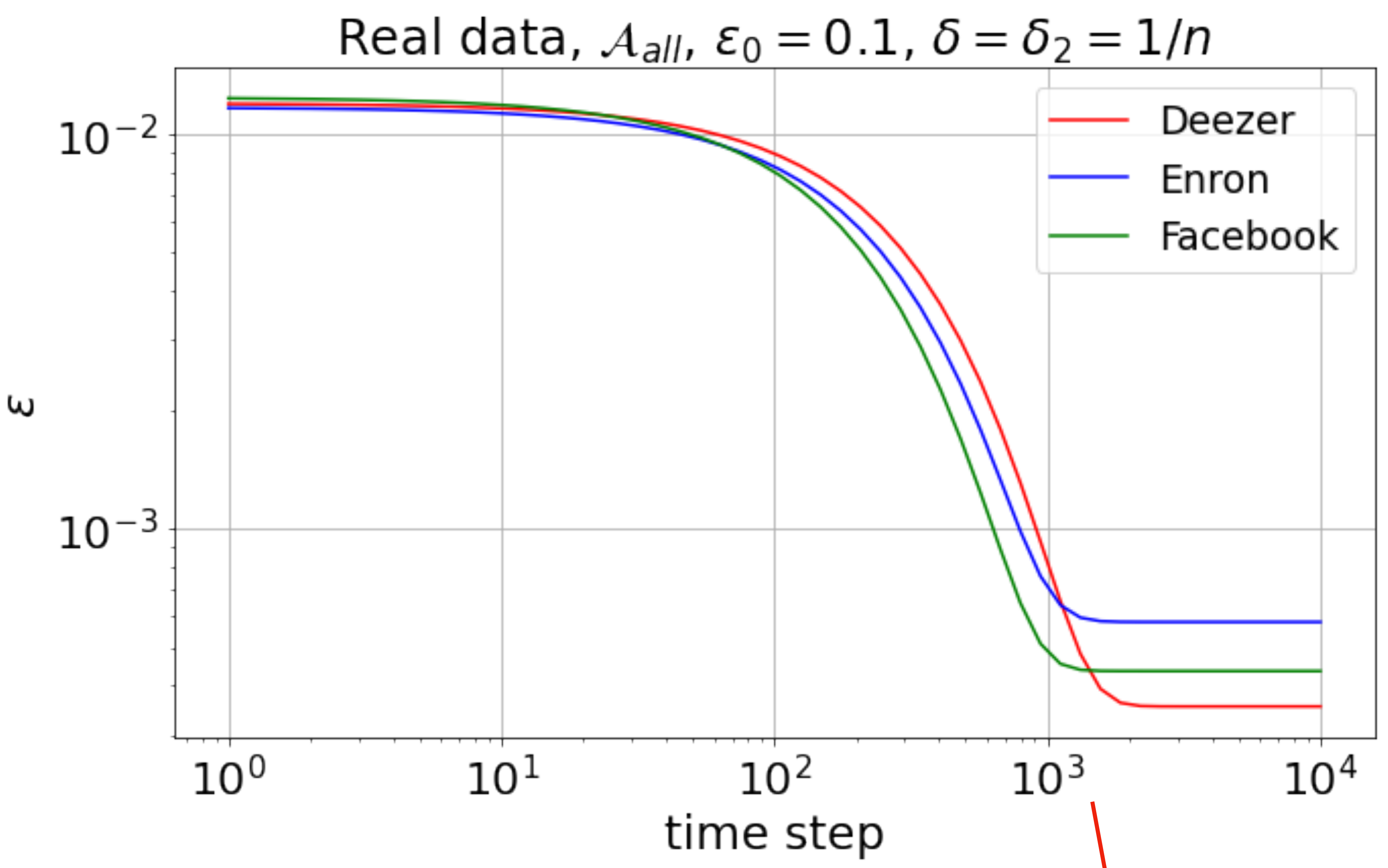
- Intuitively, the probability distribution “spreads out” with respect to time, making it harder for the adversary to guess the origin of data

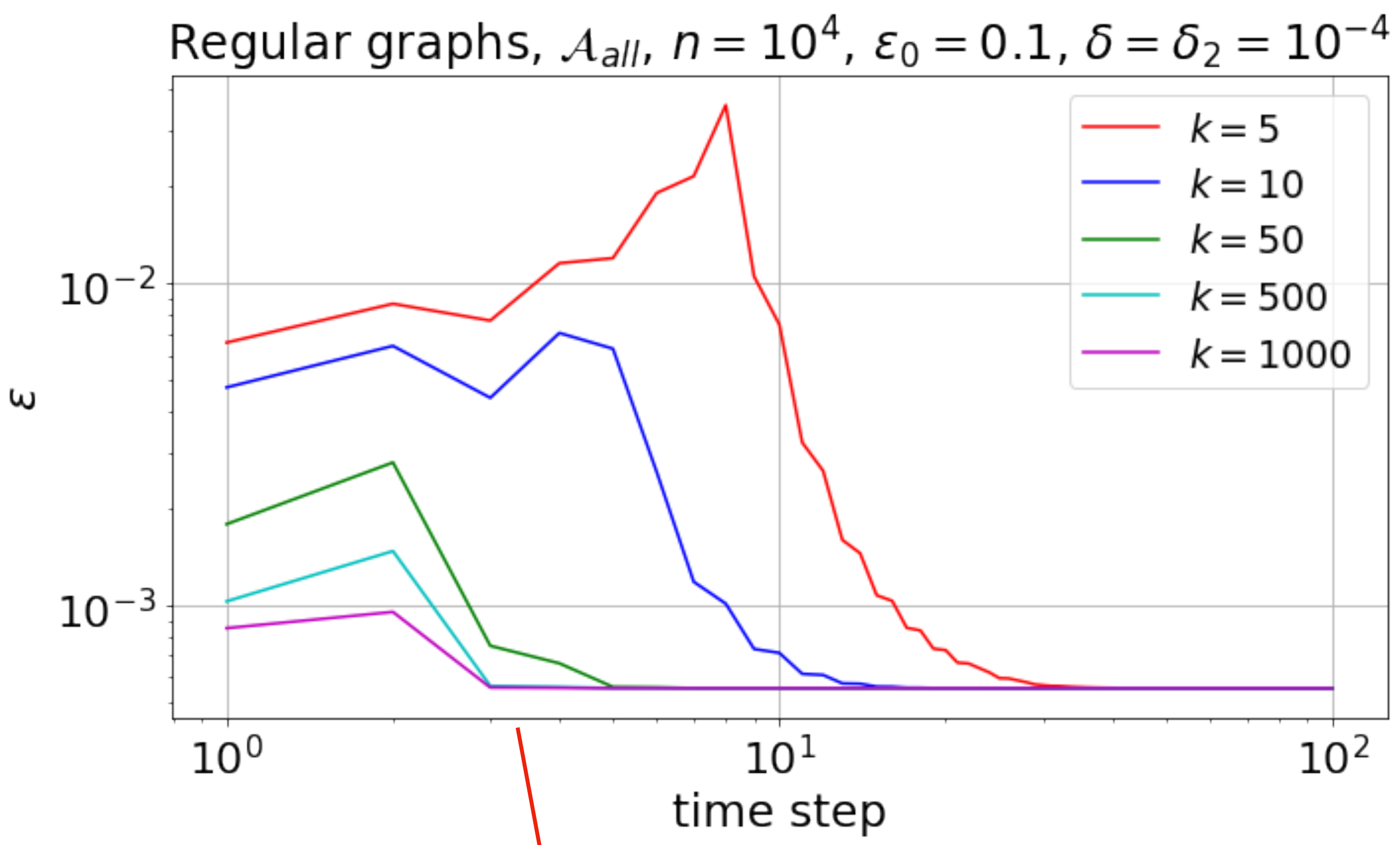


Mixing time, $\alpha^{-1} \log n$

How the privacy guarantees change with time

- Intuitively, the probability distribution “spreads out” with respect to time, making it harder for the adversary to guess the origin of data

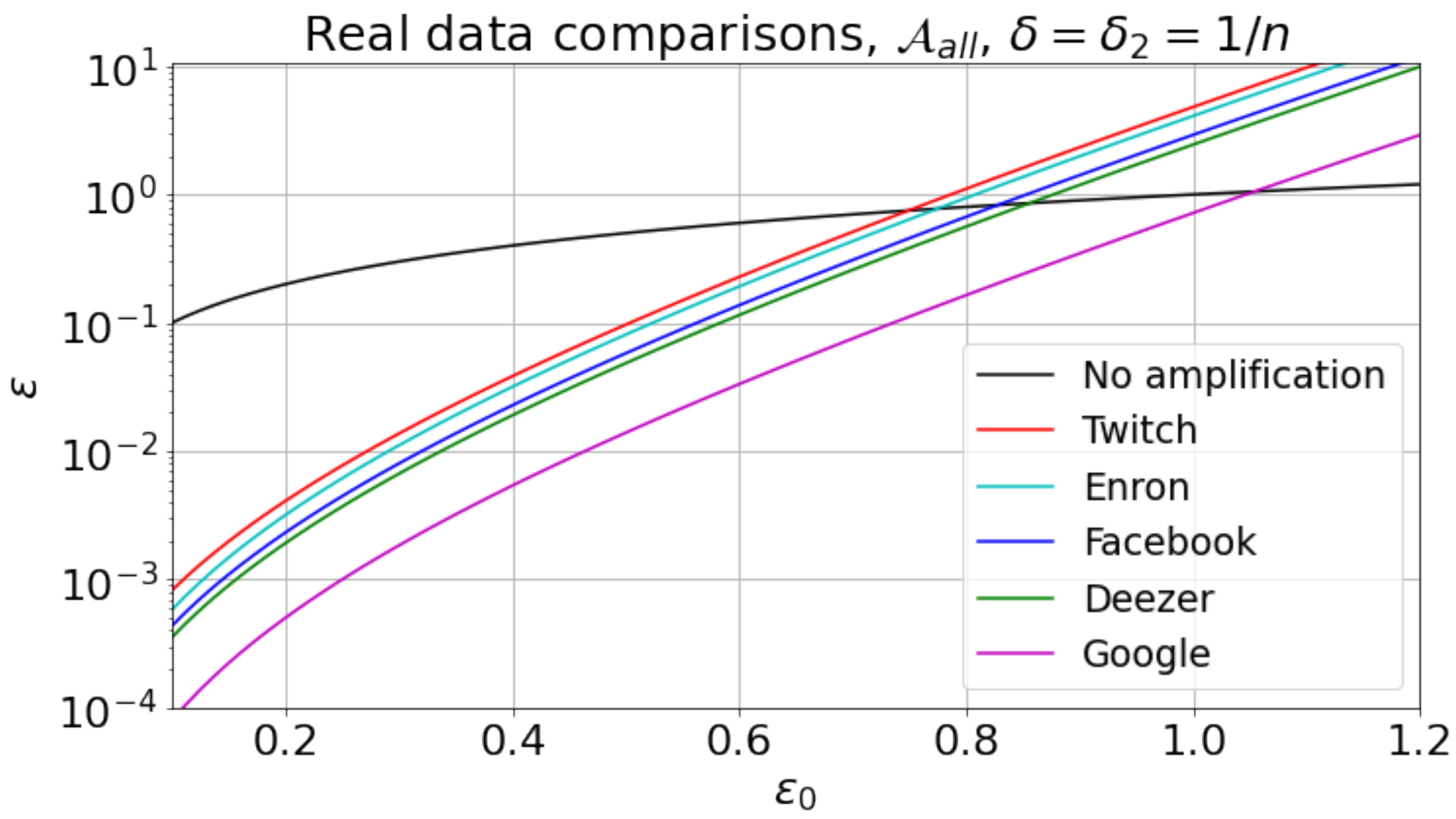




Mixing time, $\alpha^{-1} \log n$

Tracing a regular graph

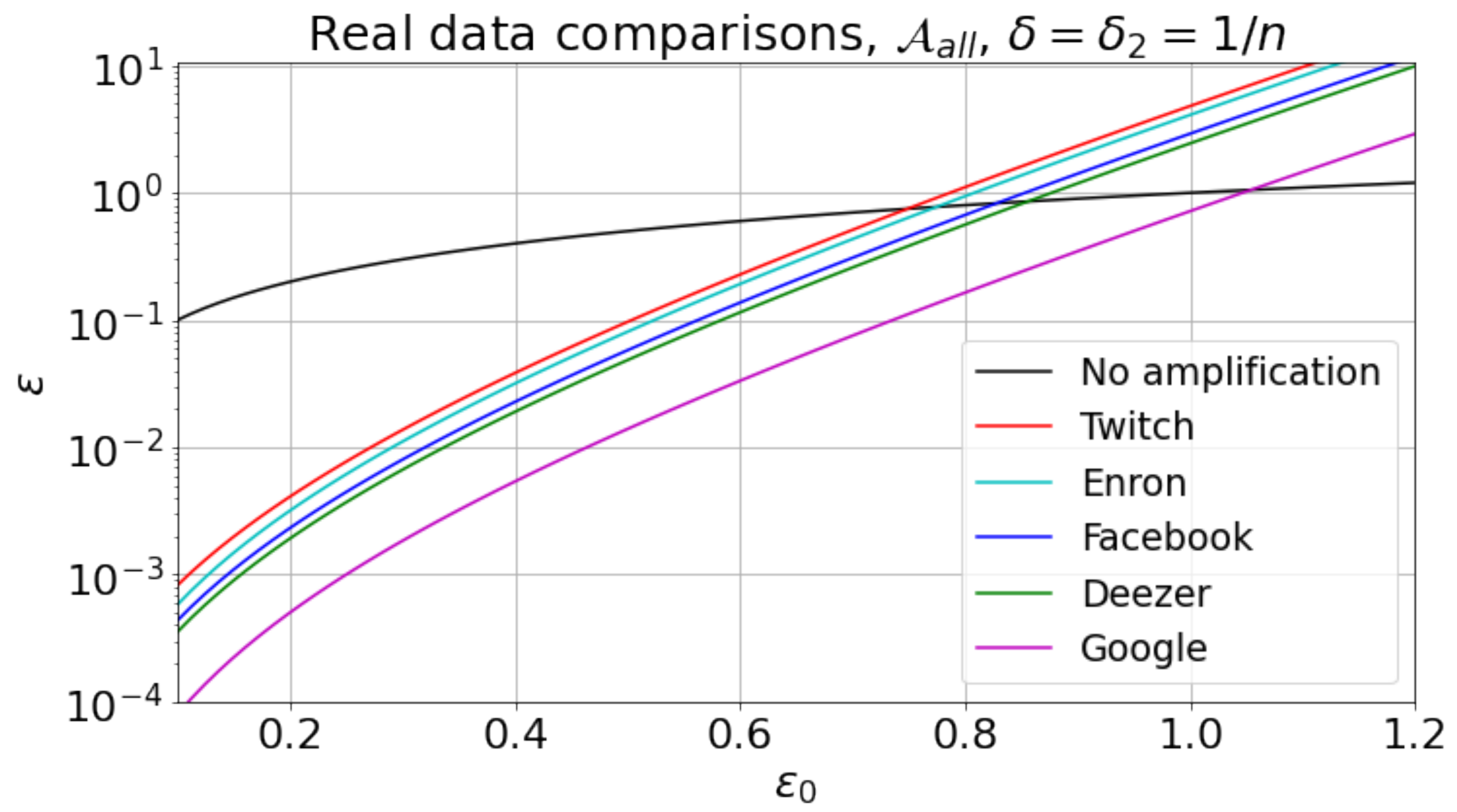
Amplification (ϵ_0 vs ϵ)



- Larger population leads to more significant amplification (Google: 856k vs Twitch: 9k)

- Amplification does not occur at large ϵ_0

Amplification (ϵ_0 vs ϵ)



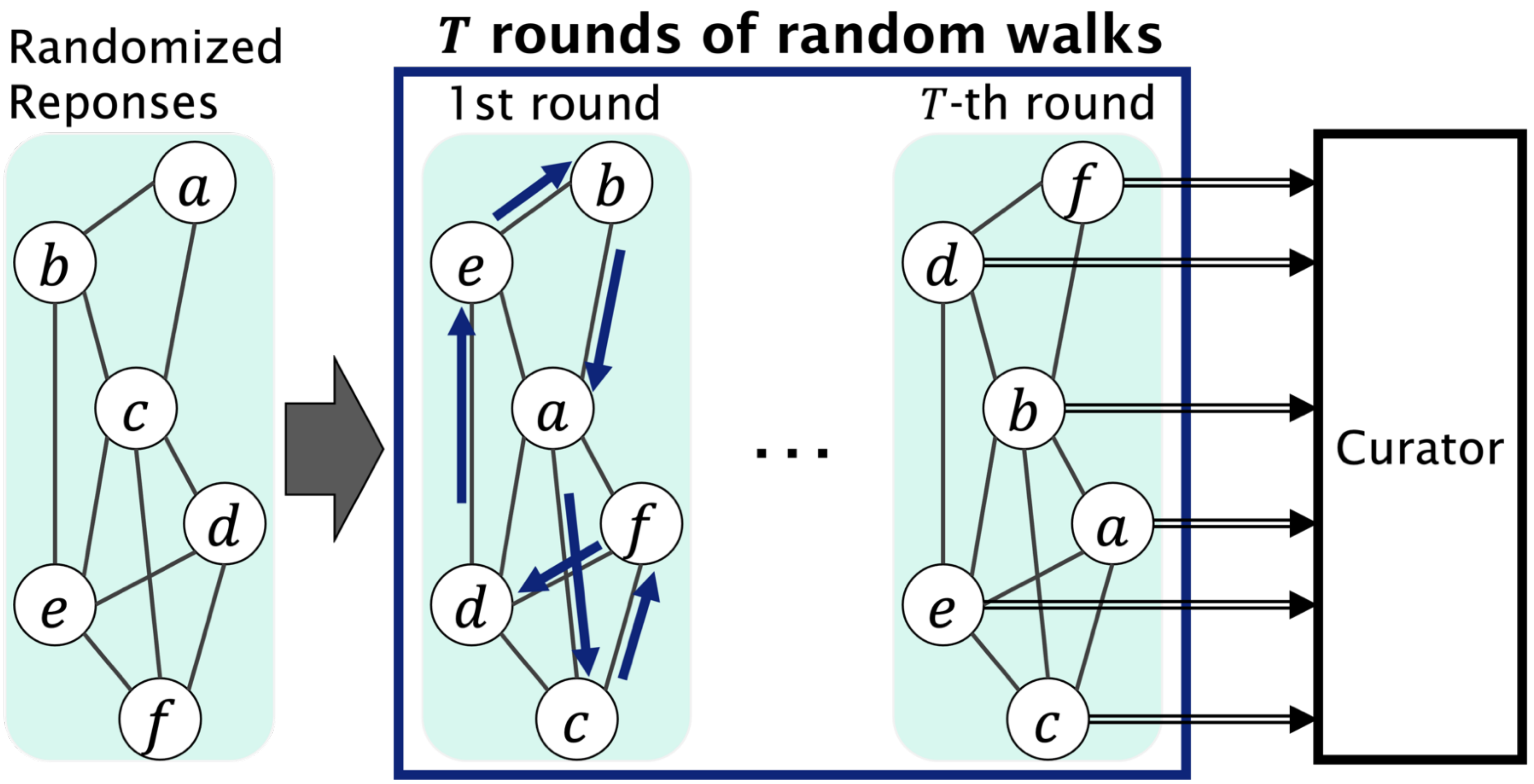
Mechanism	Privacy Amplification
No amplification [18]	ε_0
Uniform subsampling [1, 33]	$O(e^{\varepsilon_0}/\sqrt{n})$
Uniform shuffling [22]	$O(e^{3\varepsilon_0}/\sqrt{n})$
Uniform shuffling (w/ clones) [25]	$O(e^{0.5\varepsilon_0}/\sqrt{n})$
Network shuffling (ours)	$O(e^{1.5\varepsilon_0}/\sqrt{n})$

- Larger population leads to more significant amplification (Google: 856k vs Twitch: 9k)
- Amplification does not occur at large ϵ_0

- Similar rate of amplification (weaker exponential dependence)
- Could be improved with more advanced techniques

Other topics not discussed here

- “Single” protocol where user sends only one message: stronger privacy guarantees
- Tighter privacy bound for *k*-regular graph
- Private mean estimation as an application
- Threat modeling
- Please check our paper or **arXiv:2204.03919**



APPENDICES

Protocol

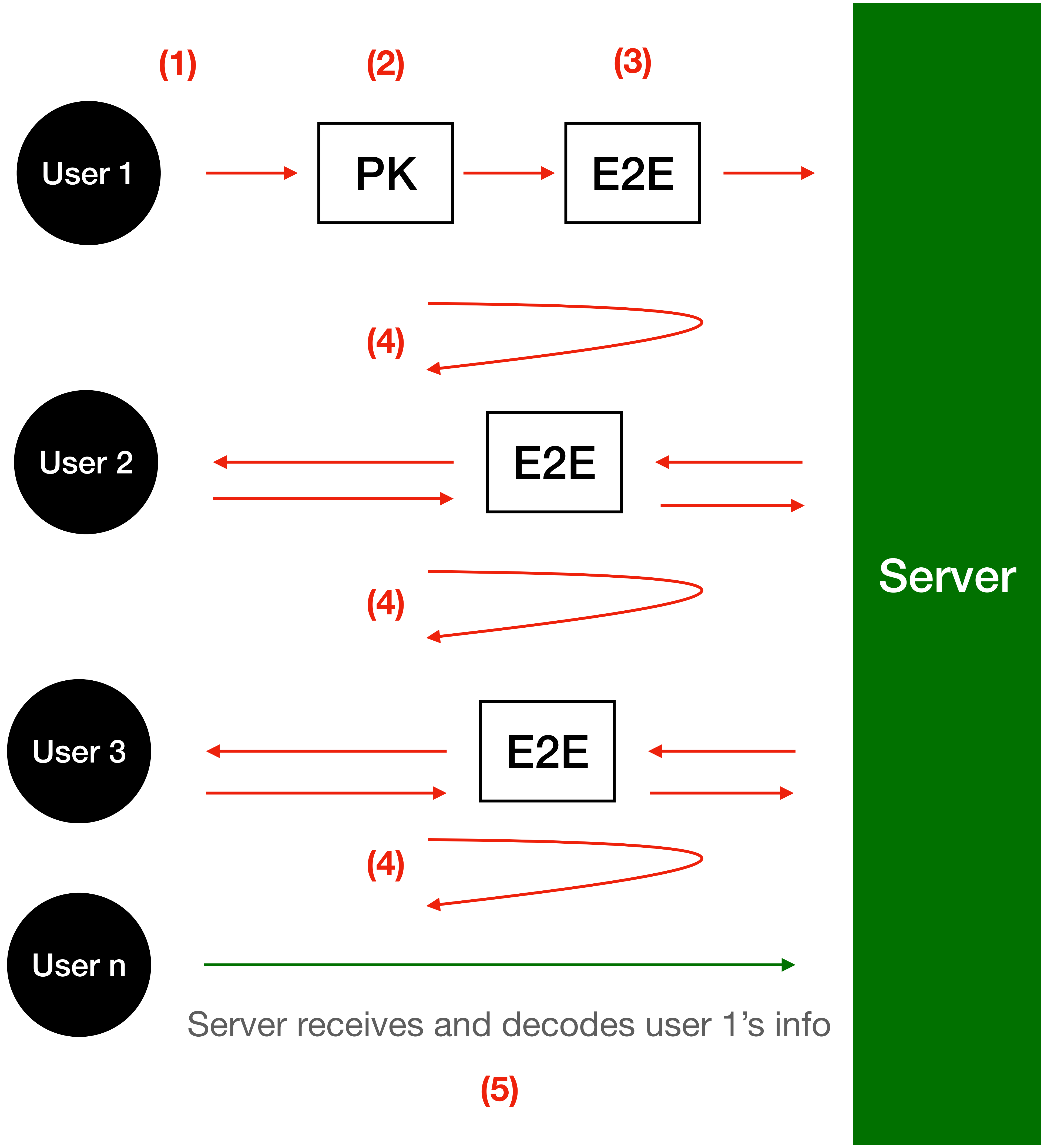
1. For each user/client, add noise to the output using local randomizer.

2. Use a public key (PK) provided by Server to encrypt the noisy output (to prevent eavesdropping by parties other than Server, e.g., other clients).

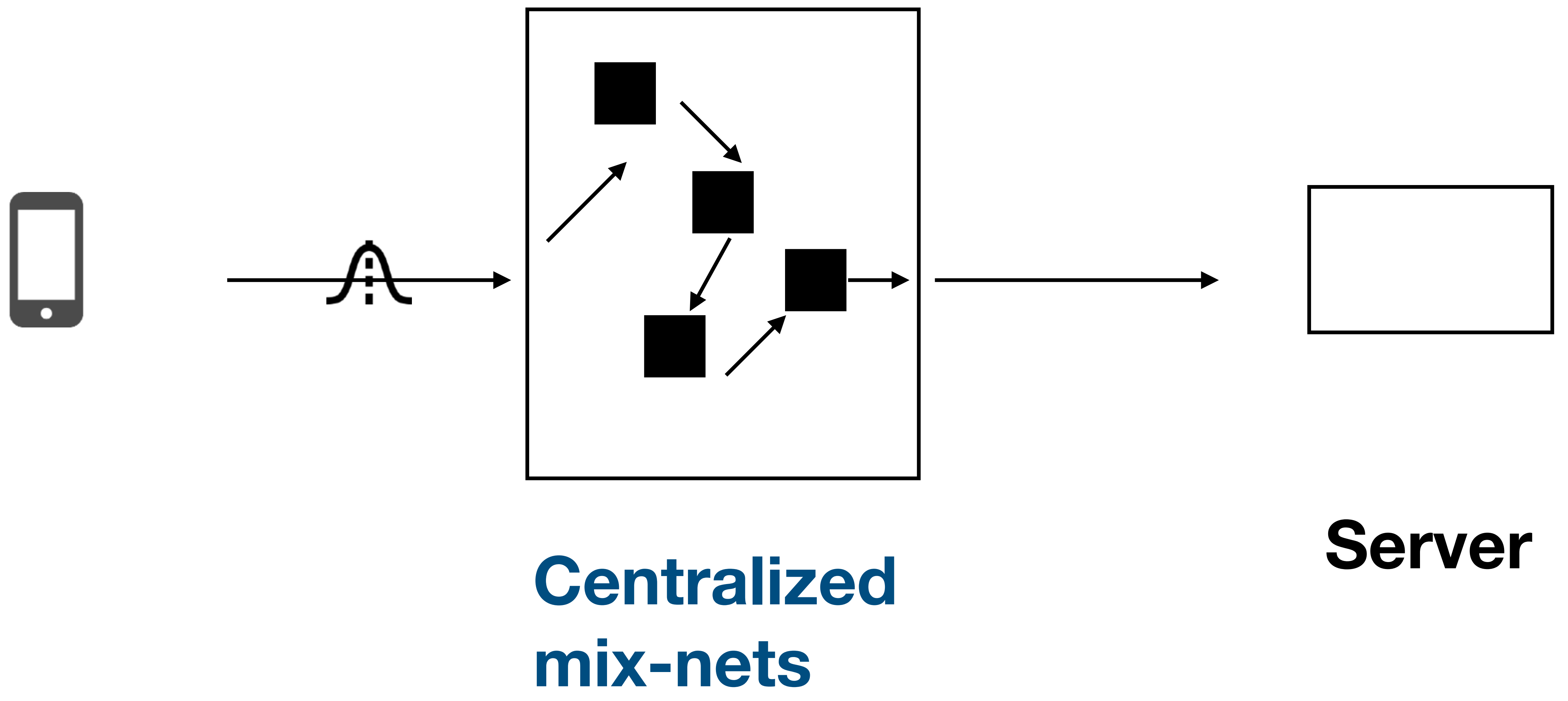
3. Communicate with other users via end-to-end encryption (to prevent eavesdropping by parties other than the receiver, e.g., Server)

4. Send to a random user the noisy output via E2E.

5. Send noisy output to server after a pre-determined number of communication rounds



Trusted shuffler implementations



- n^2 communication complexity due to cover traffic
- still need to trust extra centralized entities