

Shuffle Private Linear Contextual Bandits

Xingyu Zhou, Sayak Ray Chowdhury*

UCLA Big Data and Machine Learning Seminar

* Equal Contributions, Post-doc at Boston University

Introduction

Linear Contextual Bandits (LCB)

○ For each time $t = 1, \dots, T$

1. Observe context c_t

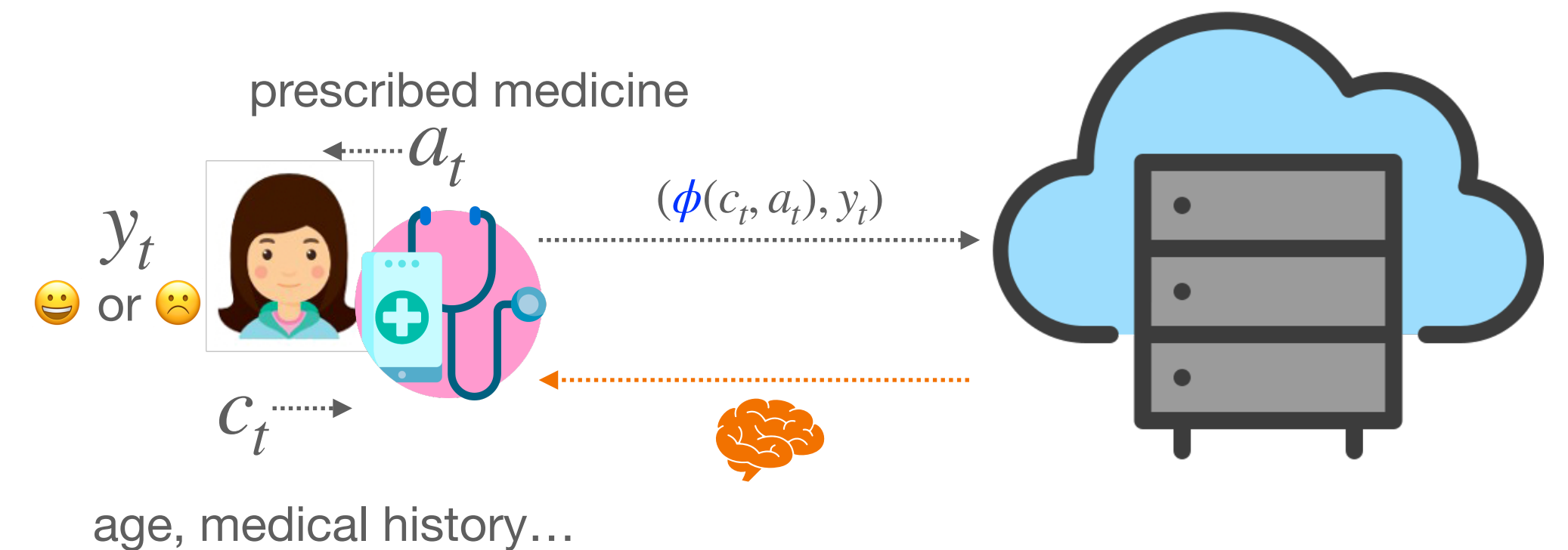
2. Prescribes action a_t

3. Receive reward $y_t = \langle \phi(c_t, a_t), \theta^* \rangle + \epsilon_t$




4. Update model

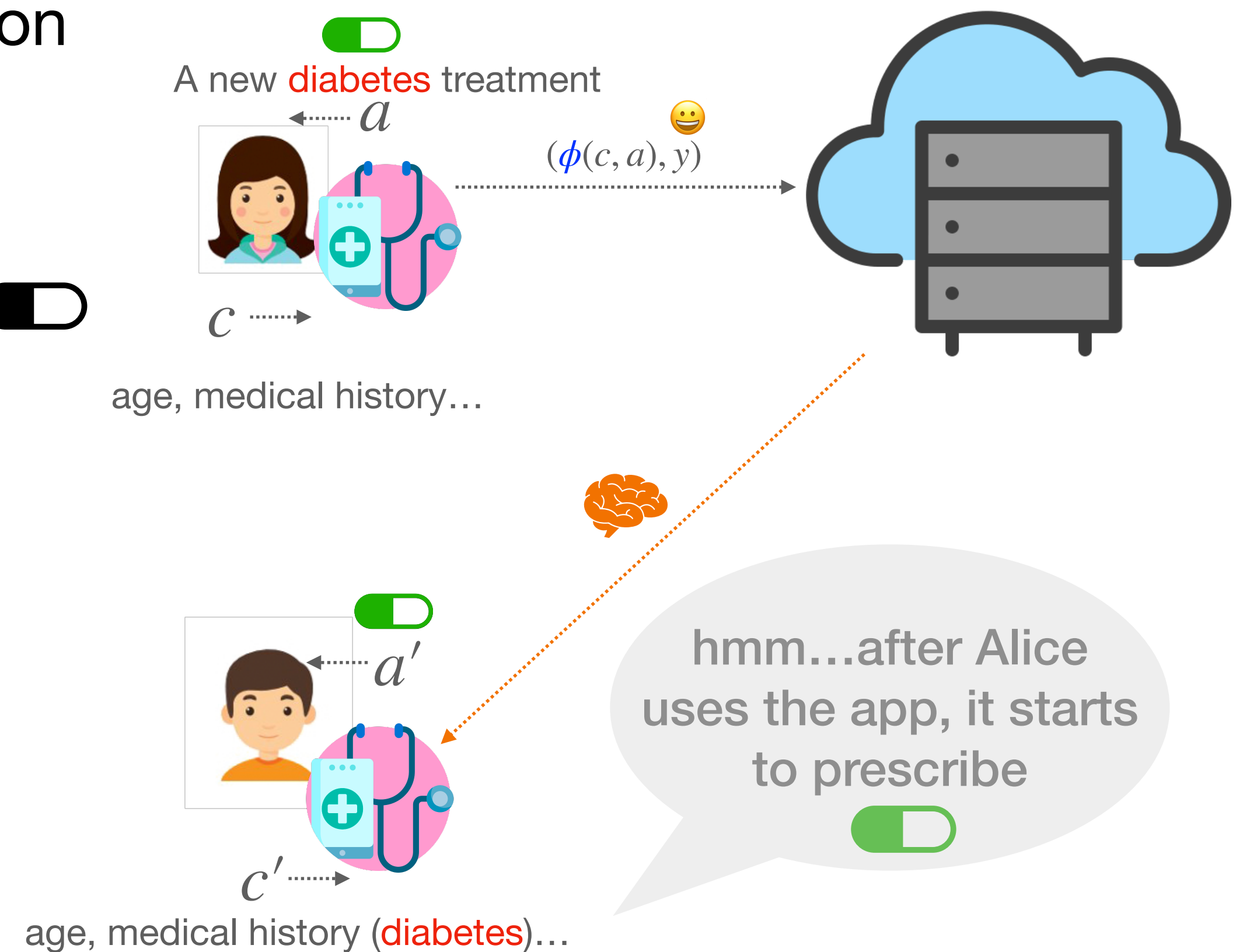
○ The goal is to minimize regret

$$\text{Reg}(T) = \sum_{t=1}^T \left[\max_a \langle \theta^*, \phi(c_t, a) \rangle - \langle \theta^*, \phi(c_t, a_t) \rangle \right]$$



Privacy Risk

- Both **context** and **reward** are sensitive information
- Standard LCB could reveal these information
 - Bob has **diabetes** and health app often prescribes 
 - Alice is a **new** user and extremely happy with 
 - Bob receives new recommendation 
 - If Bob knows Alice is the most recent user*
 - Bob's belief that Alice has diabetes **increases**



Differentially Private LCB

Central model

- Differential Privacy (DP) provides formal privacy guarantee [Dwork et al. 2006]

- Well-tuned noise added to obscure each user's contribution

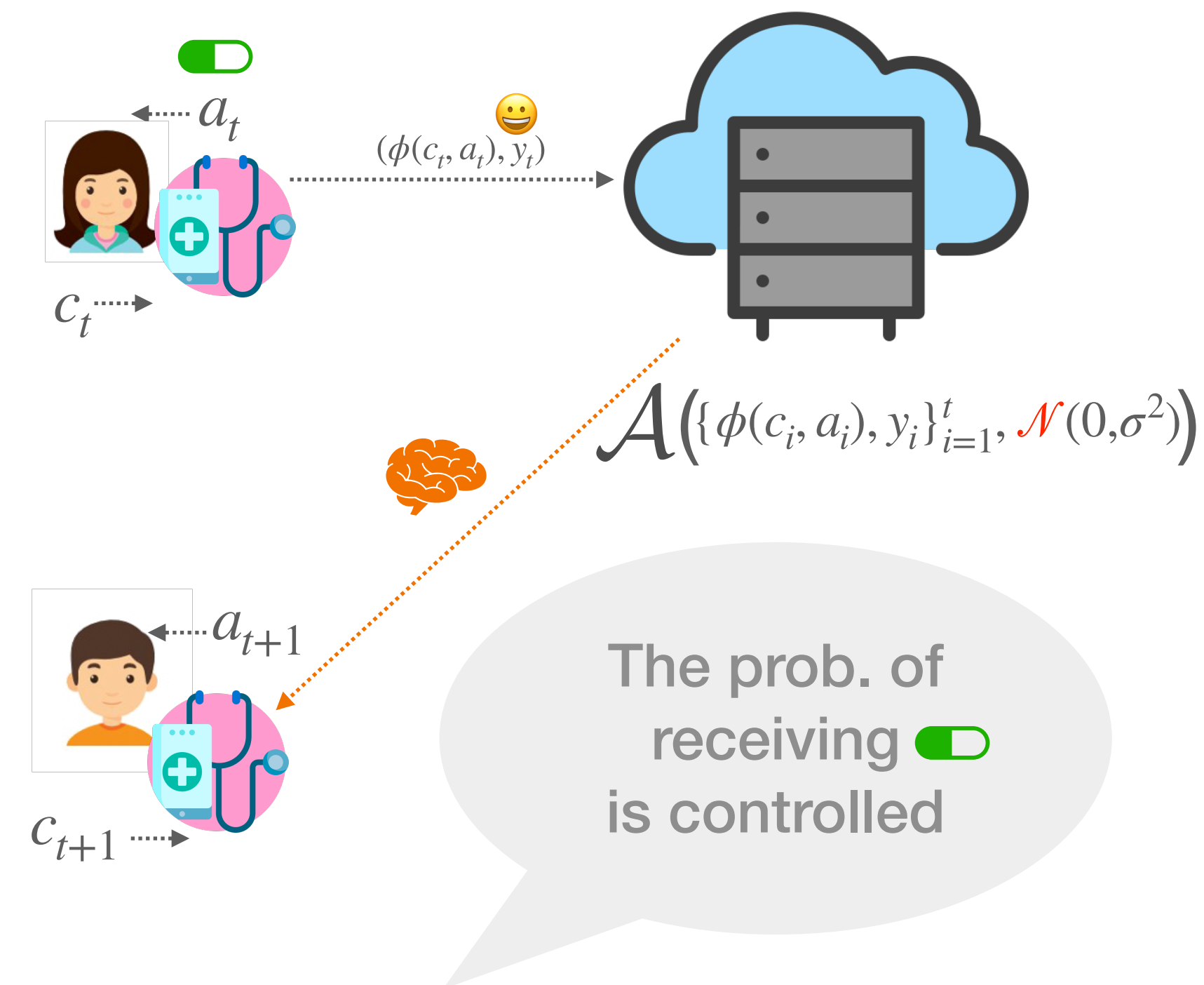
- In LCB, **central server** updates model with injected noise

 - Gaussian noise with variance $\sigma^2 = O(\log(1/\delta)/\epsilon^2)$

 - Smaller ϵ, δ , stronger privacy but worse regret

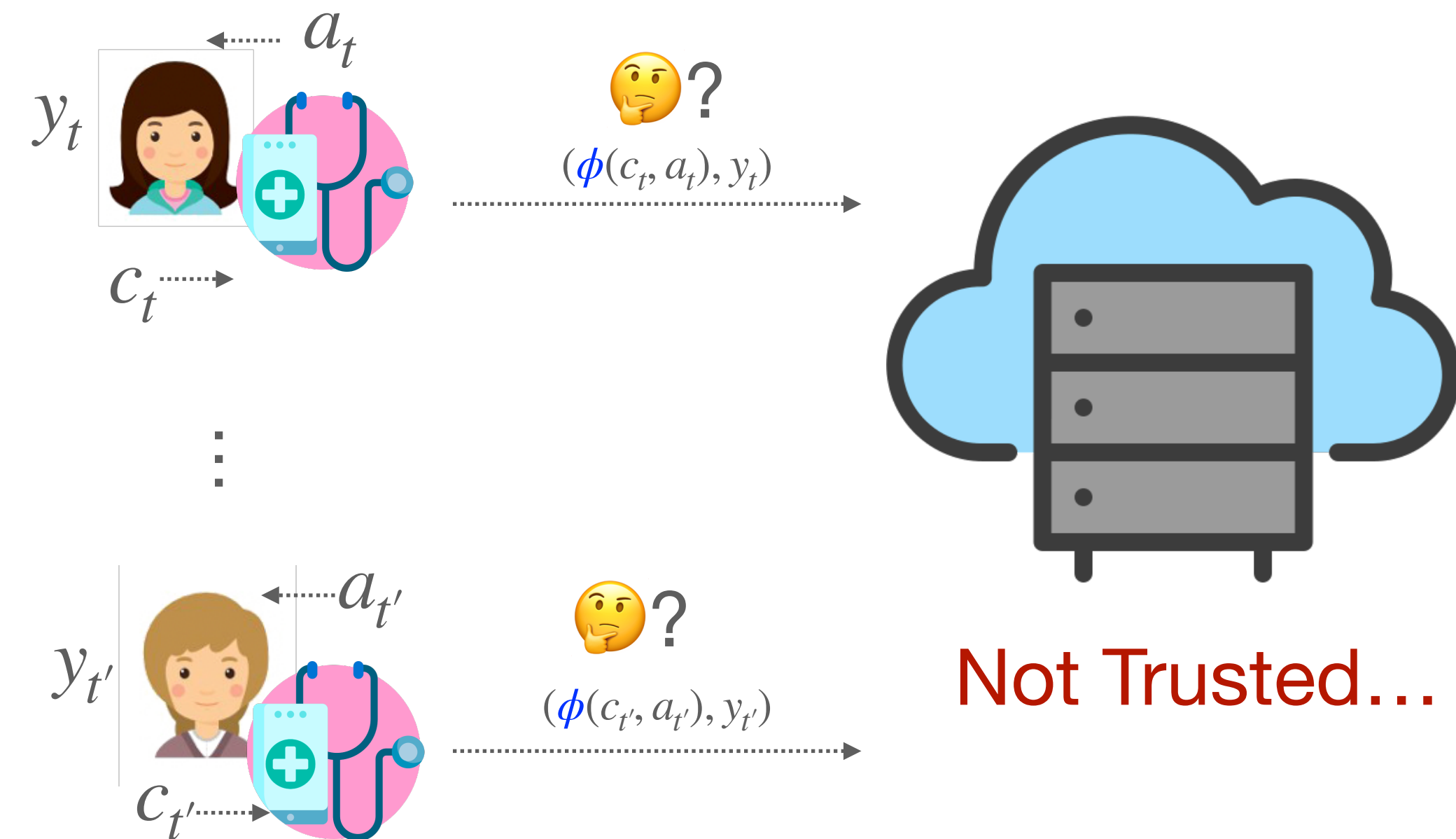
- **Privacy vs Regret.** [Shariff and Sheffet. 2018] shows that

$$\text{Regret } \tilde{O}\left(\frac{\sqrt{T}(\log(1/\delta))^{1/4}}{\sqrt{\epsilon}}\right) \text{ under central } (\epsilon, \delta)\text{-DP}^*$$



Another Privacy Risk

- Both **context** and **reward** are sensitive information
- What if central server is **not** trustworthy?
 - *Will it follow the right DP mechanism...?*
 - *Will it use my data for other use cases...?*
 - *Will it be attacked by an adversary...?*
- **Hence**, users may **not** be willing to share their raw data
 - Context via $\phi(c_t, a_t)$
 - Reward y_t

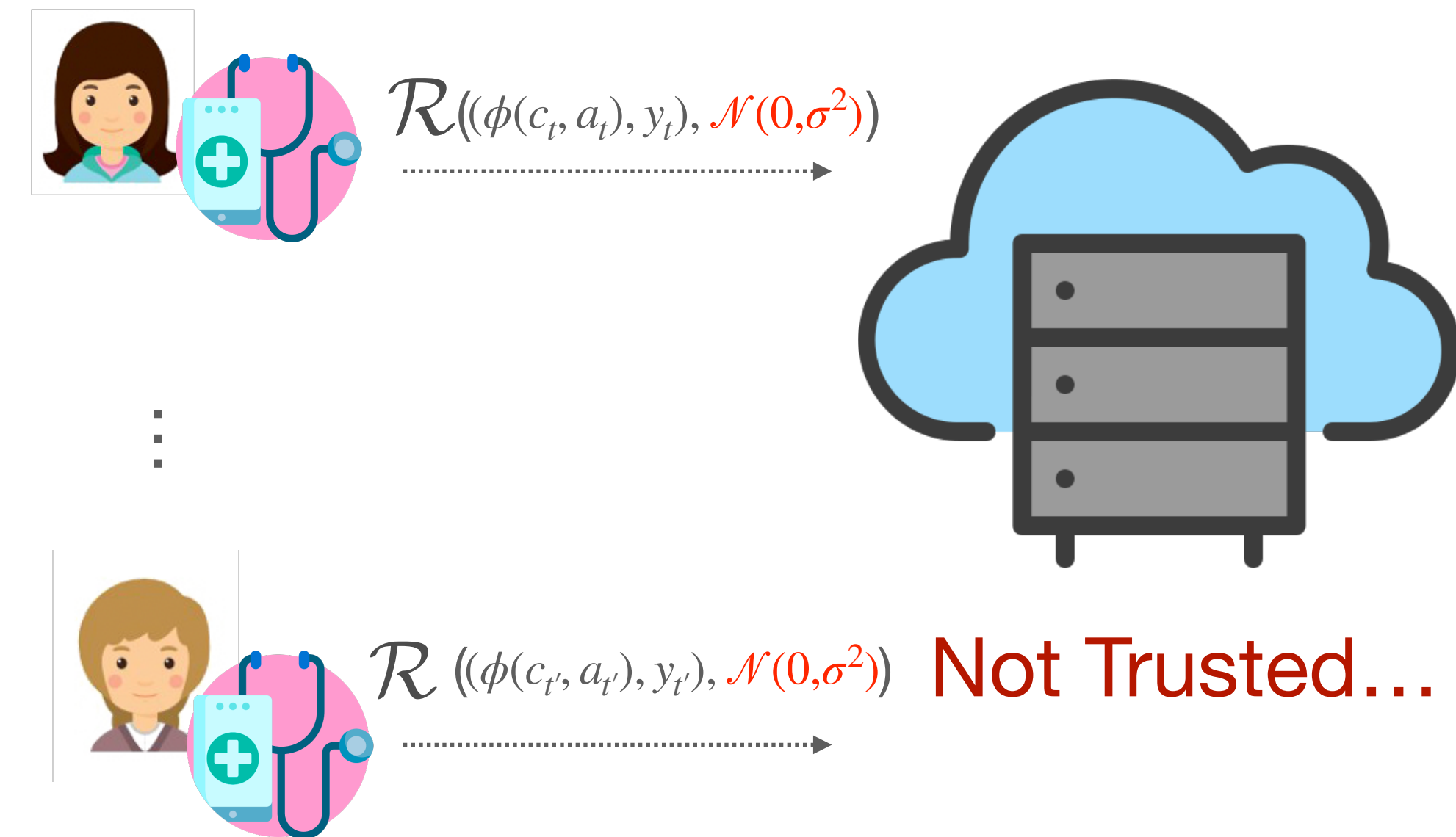


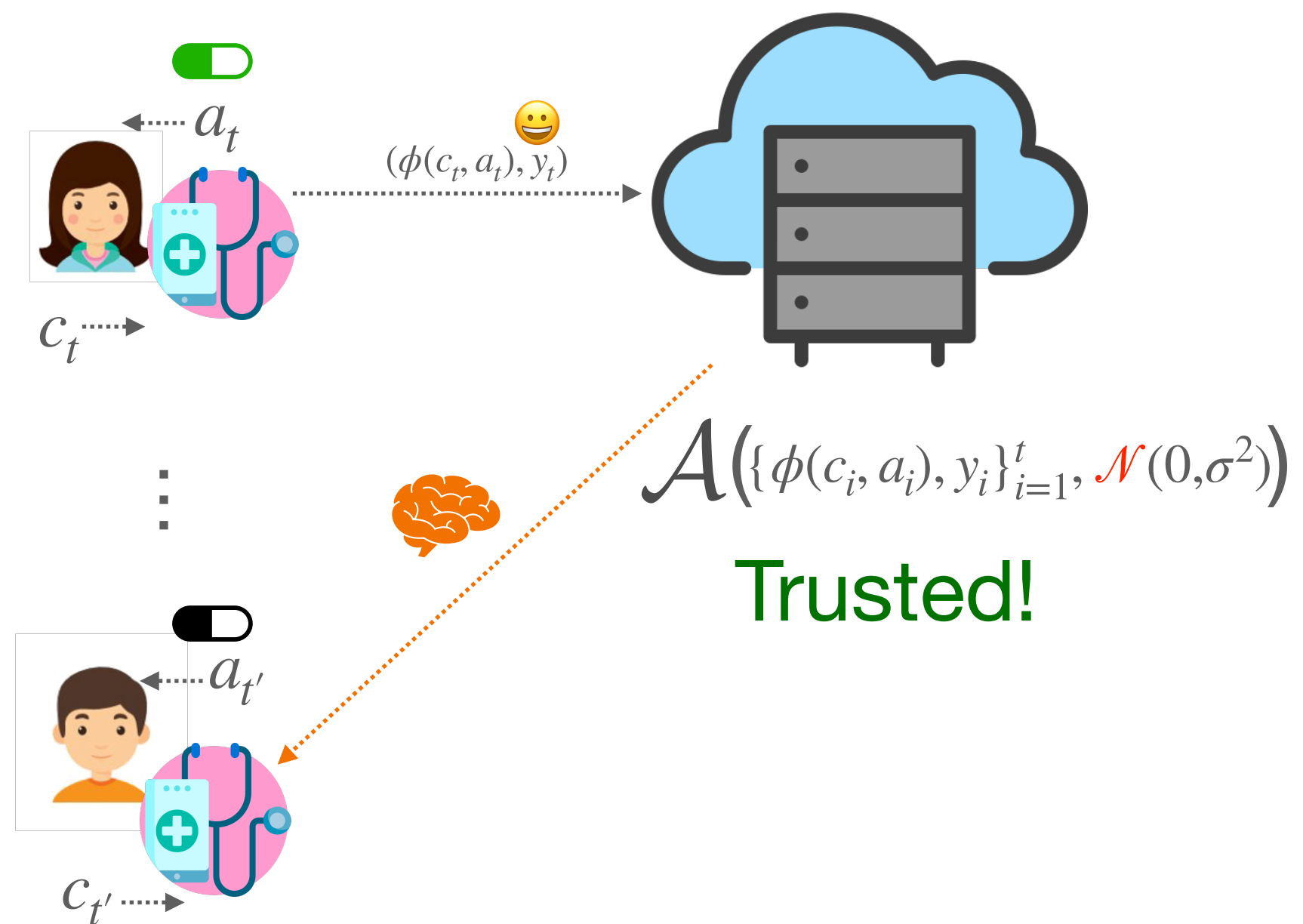
Differentially Private LCB

Local model

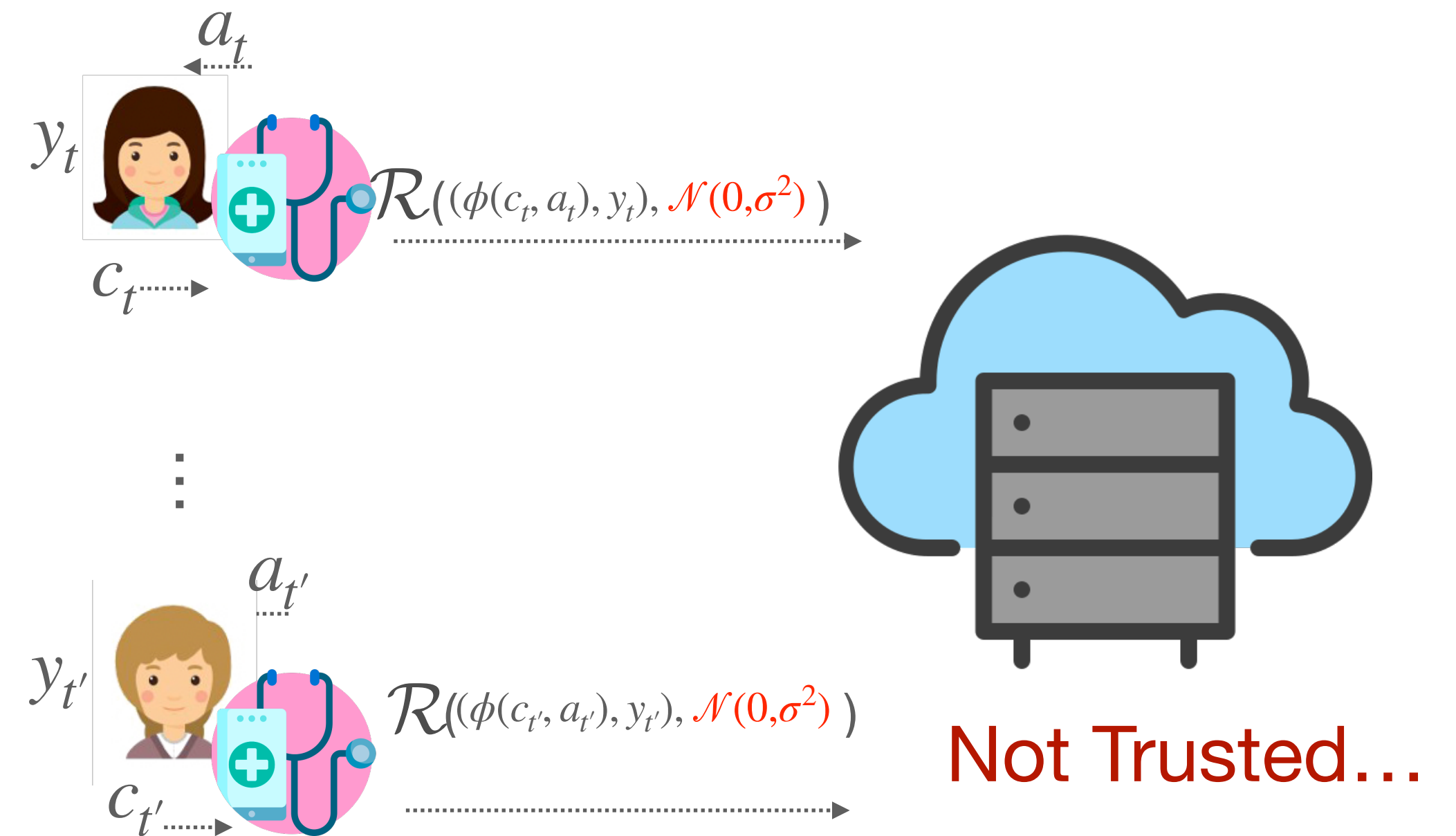
- Each user injects noise before sending data
 - By post-processing, local DP implies central DP
- In LCB, **each user** applies local randomizer \mathcal{R}
 - Gaussian noise with variance $\sigma^2 = O(\log(1/\delta)/\epsilon^2)$
 - Smaller ϵ, δ , stronger privacy but worse regret
- Privacy vs Regret.** [Zheng et al. 2020] shows that

$$\text{Regret } \tilde{O}\left(\frac{T^{3/4}(\log(1/\delta))^{1/4}}{\sqrt{\epsilon}}\right) \text{ under local } (\epsilon, \delta)\text{-DP}^*$$





$$\text{Regret } \tilde{O}\left(\frac{\sqrt{T}(\log(1/\delta))^{1/4}}{\sqrt{\epsilon}}\right) \text{ under central } (\epsilon, \delta)\text{-DP}$$

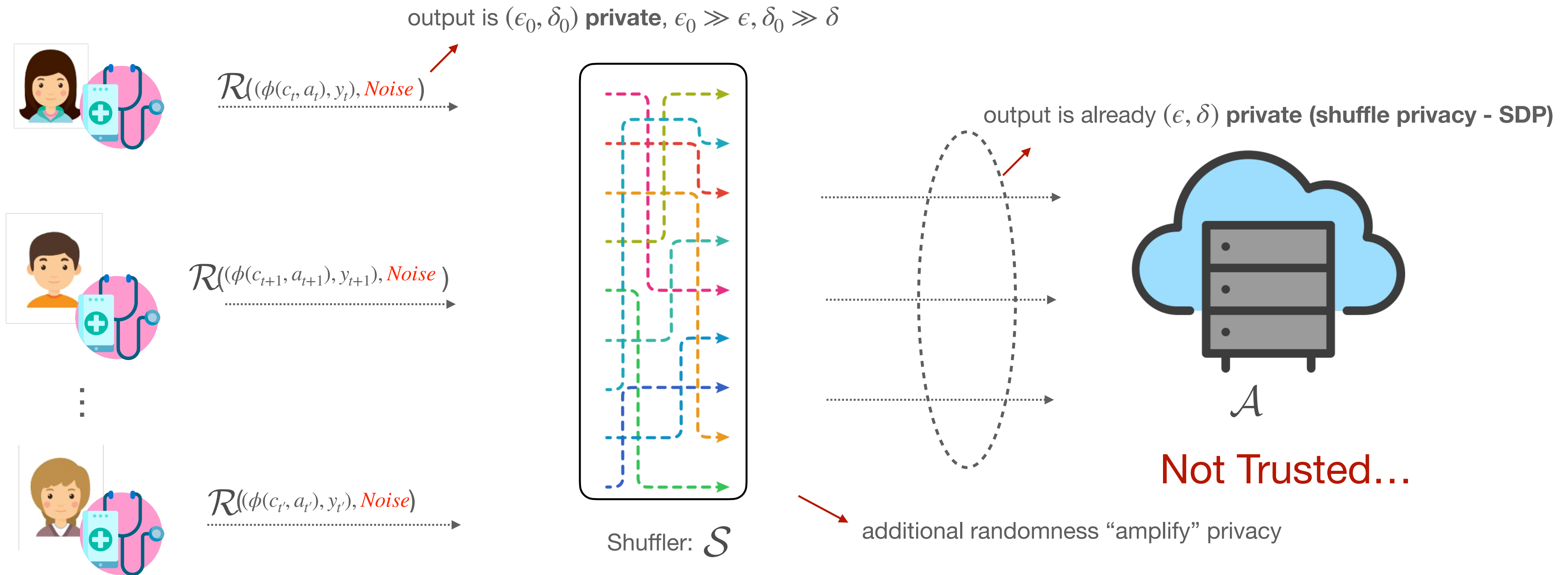


$$\text{Regret } \tilde{O}\left(\frac{T^{3/4}(\log(1/\delta))^{1/4}}{\sqrt{\epsilon}}\right) \text{ under local } (\epsilon, \delta)\text{-DP}$$

Can one achieve a better regret even without a trusted server?

Yes!

Contribution



1. Propose a generic private LCB algorithm with black-box protocol $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$
2. Two instantiation of \mathcal{P} guarantee *shuffle privacy* with regret $\tilde{O}(T^{3/5})$
3. For the case of returning users, our regret can **match** the one under central model, i.e, $\tilde{O}(T^{2/3})$

Related Work

Shuffle DP protocols & app. in SGD

- Shuffle DP protocols

- Practical system [Bittau et al. 2017 ...]
- Shuffle protocols for bounded sum [Cheu et al. 2021[✨], Cheu et al. 2019, Balle et al. 2020, Ghazi et al. 2020 ...]
 - Sum of n numbers in $[0,1]$, shuffler enables (ϵ, δ) -SDP with error $\tilde{O}(1/\epsilon)$
- General “privacy amplification” bounds [Feldman et al. 2021[✨], Erlingsson et al. 2019, Balle et al. 2019 ...]
 - Shuffling of n ϵ_0 -DP locally randomized data, yields (ϵ, δ) -SDP with $\epsilon = \tilde{O}(\epsilon_0/\sqrt{n})$ if $\epsilon_0 \leq 1^*$

- Applications in private SGD

- Both ERM and SCO [Girgis et al. 2021, Lowy and Razaviyayn 2021, Cheu et al. 2021 ...]
 - Shuffler enables SDP with the **same** convergence rate as in central DP

Related Work

Shuffle DP in bandit learning

- Shuffle DP in MAB [Tenebaum et al. 2021]
 - A batch-variant arm elimination algorithm
 - Guarantee (ϵ, δ) -SDP with **additive** privacy cost $\frac{K \log T \sqrt{\log(1/\delta)}}{\epsilon}$
 - Central $(\epsilon, 0)$ -DP — additive cost $\frac{K \log T}{\epsilon}$; Local $(\epsilon, 0)$ -DP — multiplicative factor $1/\epsilon^2$
- Shuffle DP in linear contextual bandits
 - In addition to rewards, contexts also need protection
 - One concurrent and independent work [Garcelon et al. 2021]
 - More complicated algorithm; A gap exists in their regret analysis*
 - The shuffle privacy guarantee only holds for $\epsilon \ll 1$

Background

Shuffle Differential Privacy

Standard SDP

- **Neighboring datasets.** $D, D' \in \mathcal{D}^n$ are *neighboring* if they only differ in one user's data D_i

Def. Differential Privacy [Dwork et al. 2006]

For $\epsilon, \delta > 0$, a randomized mechanism \mathcal{M} satisfies (ϵ, δ) -DP is for **all** neighboring datasets D, D' and **all** events \mathcal{E} in the range of \mathcal{M}

$$\mathbb{P}[\mathcal{M}(D) \in \mathcal{E}] \leq e^\epsilon \cdot \mathbb{P}[\mathcal{M}(D') \in \mathcal{E}] + \delta$$

- **Standard shuffle DP.** The output of the shuffler is private, i.e., $(\mathcal{S} \circ \mathcal{R}^n) := \mathcal{S}(\mathcal{R}(D_1), \dots, \mathcal{R}(D_n))$

Def. Shuffle Diff. Privacy [Cheu et al. 2019]

Let $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ be a protocol for n users. Then, \mathcal{P} satisfies (ϵ, δ) -SDP if the mechanism $(\mathcal{S} \circ \mathcal{R}^n)$ satisfies (ϵ, δ) -DP

-  Recall that shuffling amplifies privacy by \sqrt{n}

Shuffle Differential Privacy

SDP in Bandits

- **Divide users into batch.** Run a standard protocol for each batch $m \in [M]$ with size n_m
- **Composite mechanism.** $\mathcal{M}_{\mathcal{P}} = (\mathcal{S} \circ \mathcal{R}^{n_1}, \dots, \mathcal{S} \circ \mathcal{R}^{n_M})$
 - Each $(\mathcal{S} \circ \mathcal{R}^{n_m})$ operates on n_m users' data \mathcal{D}^{n_m}
 - Each data point in LCB is $(\phi(c_i, a_i), y_i)$

Def. SDP in Bandits

An M -batch shuffle protocol \mathcal{P} is (ϵ, δ) -SDP if $\mathcal{M}_{\mathcal{P}}$ satisfies (ϵ, δ) -DP

-  If users are *unique*, it suffices to show each $(\mathcal{S} \circ \mathcal{R}^{n_m})$ satisfies (ϵ, δ) -DP

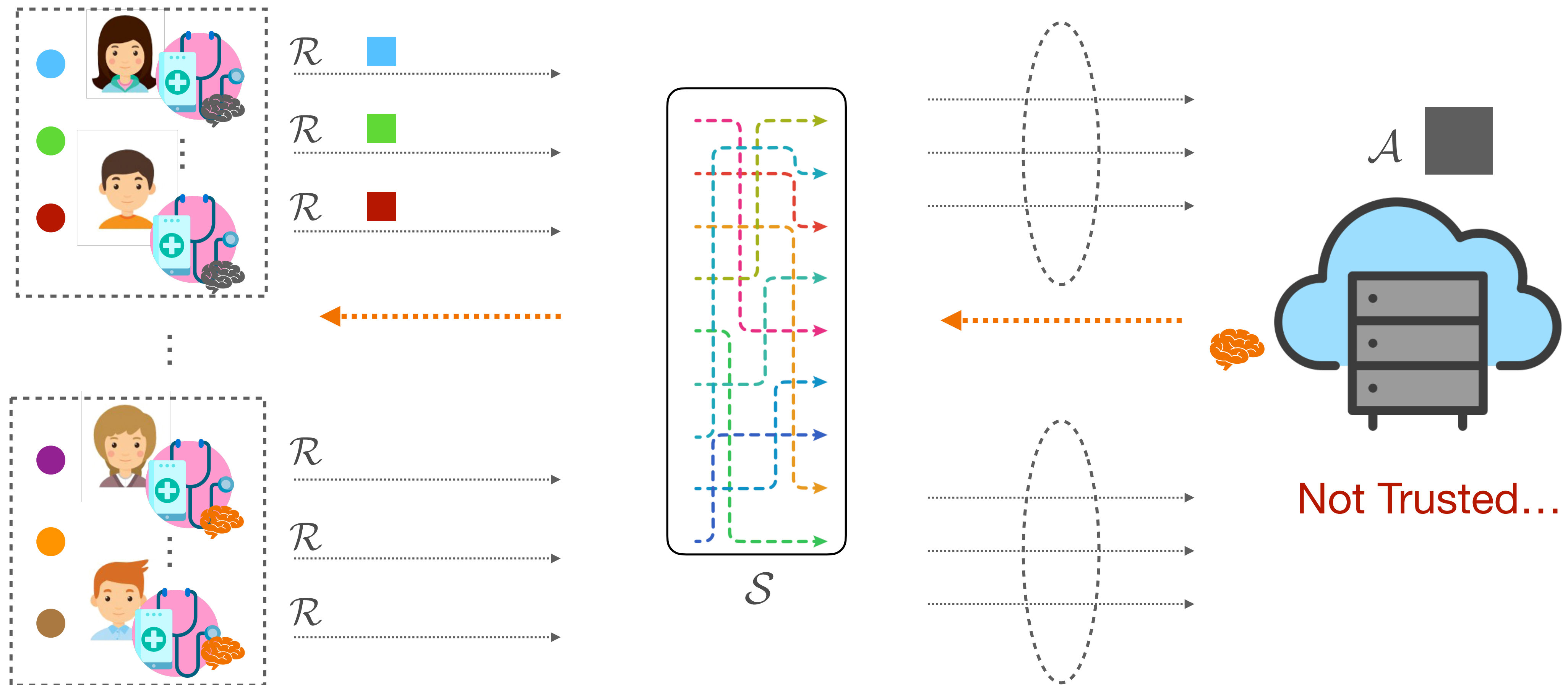
This is assumed in all previous private bandit works. We will discuss how to handle returning users later

by simple parallel-composition

Our Algorithm

A Generic Private LinUCB

Illustration



A Generic Private LinUCB

Alg. Shuffle Private LinUCB

Initialize: batch size B , statistics $V_0 = \lambda I_d$, $u_0 = 0$, initial parameter estimate $\hat{\theta}_0 = 0$

For local user $t = 1, \dots, T$ do

// user-app interaction

Observe user context c_t and prescribes action via $a_t \in \operatorname{argmax}_{a \in \mathcal{X}} \langle \phi(c_t, a), \hat{\theta}_{m-1} \rangle + \beta_{m-1} \|\phi(c_t, a)\|_{V_{m-1}^{-1}}$

User generates reward y_t

// local randomizer

Send randomized messages $M_{t,1} = R_1(\phi(c_t, a_t)y_t)$ and $M_{t,2} = R_2(\phi(c_t, a_t)\phi(c_t, a_t)^\top)$ to the shuffler

If $t = mB$ then

// shuffler

Set batch end-time $t_m = t$

Randomly permutes per-batch messages and send to central server, $Y_{m,i} = S_i(\{M_{\tau,i}\}_{t_{m-1}+1 \leq \tau \leq t_m}), i = 1, 2$

// central server

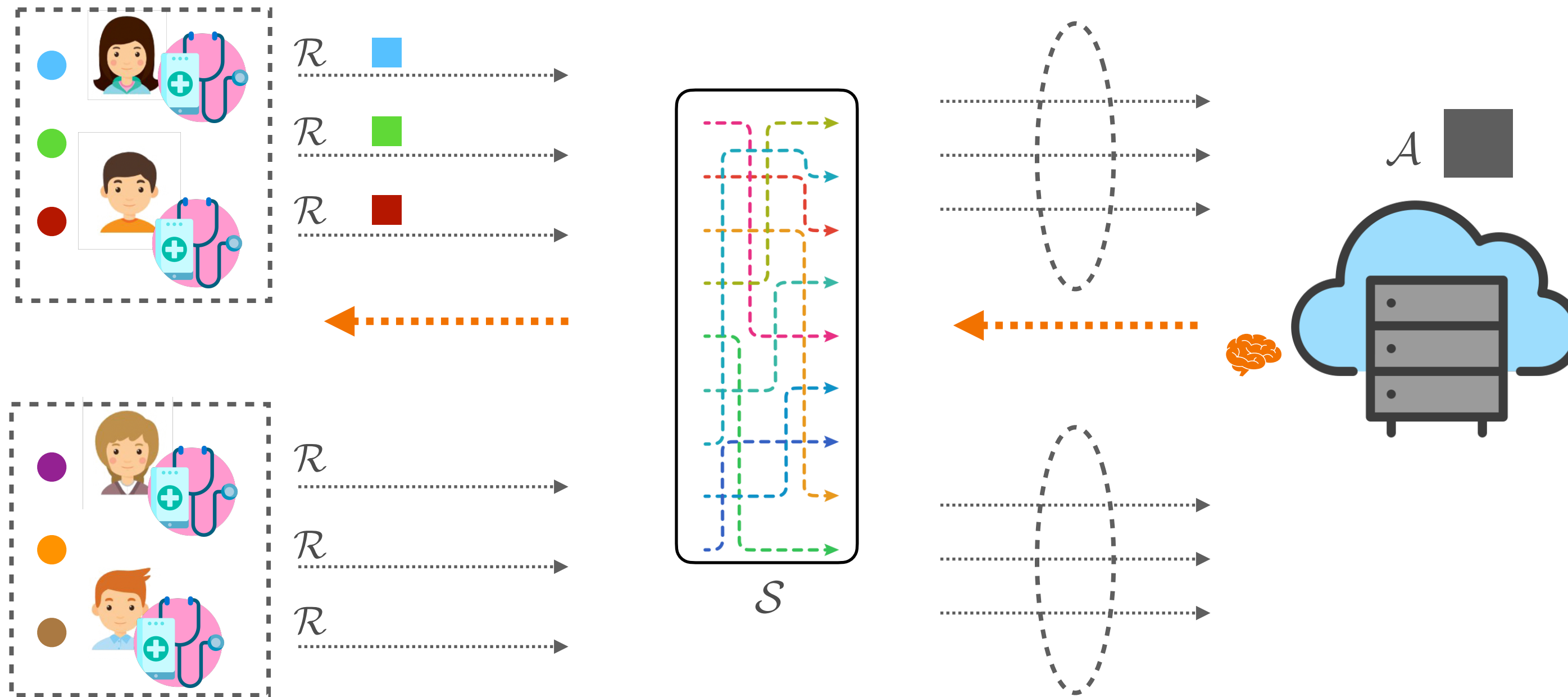
Compute *per-batch* statistics $\tilde{u}_m = A_1(Y_{m,1})$ and $\tilde{V}_m = A_2(Y_{m,2})$

Update statistics $u_m = u_{m-1} + \tilde{u}_m$ and $V_m = V_{m-1} + \tilde{V}_m$

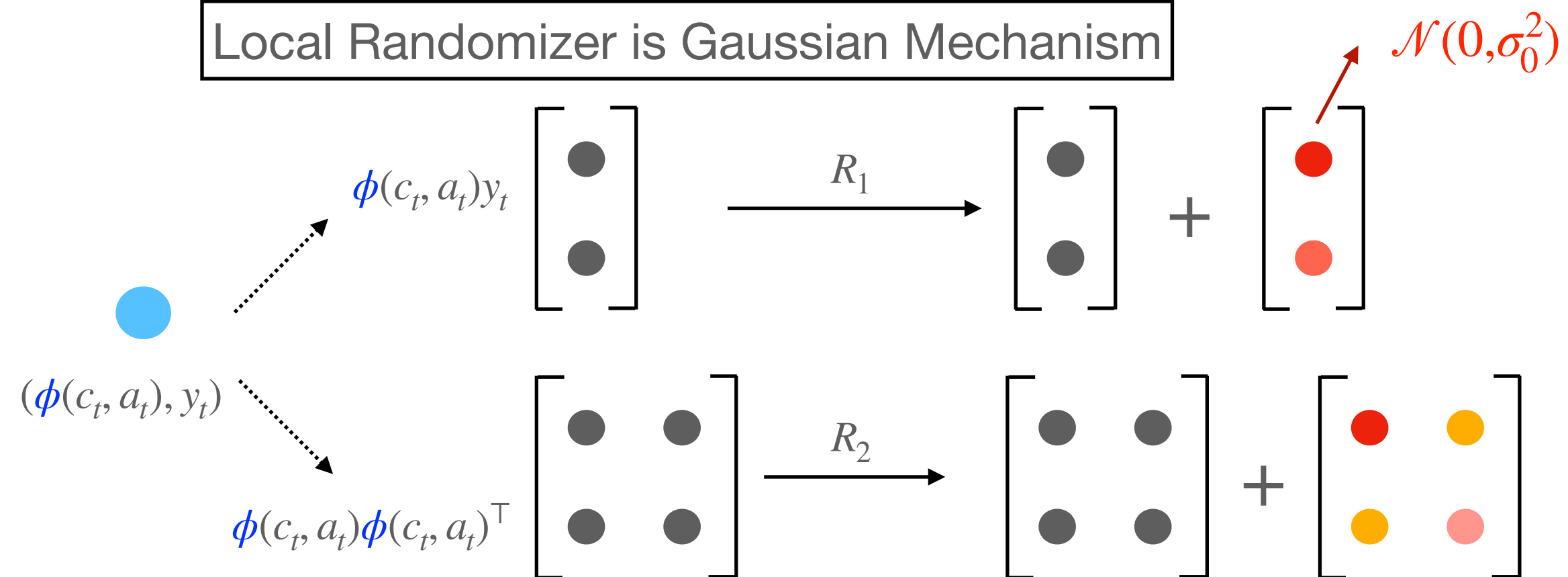
Update estimate $\hat{\theta}_m = V_m^{-1}u_m$, send new model $(\hat{\theta}_m, V_m)$ to users and increase $m = m + 1$

SDP via LDP Amplification

Amplification of Gaussian Mechanism



Local Randomizer is Gaussian Mechanism



Analyzer is a simple aggregation

$$\tilde{u} = \begin{bmatrix} \text{red dot} \\ \text{pink dot} \end{bmatrix} + \dots + \begin{bmatrix} \text{green dot} \\ \text{orange dot} \end{bmatrix}$$

$$\tilde{V} = \begin{bmatrix} \text{red dot} & \text{yellow dot} \\ \text{yellow dot} & \text{pink dot} \end{bmatrix} + \dots + \begin{bmatrix} \text{blue dot} & \text{pink dot} \\ \text{yellow dot} & \text{grey dot} \end{bmatrix}$$

Performance

SDP via Amplification

Theorem

Fix batch size B and $\epsilon \in \left(0, \sqrt{\frac{\log(2/\delta)}{B}}\right)$. Let local Gaussian mechanism choose noise $\sigma_0 = \tilde{O}(1/(\epsilon\sqrt{B}))$. Then we have

↗ LDP with $\epsilon_0 = \epsilon\sqrt{B}$

- **(Privacy)** Our algorithm is $O(\epsilon, \delta)$ -SDP
- **(Regret)** Set $B = O(T^{3/5})$, with a high probability, our algorithm achieves $\tilde{O}\left(\frac{T^{3/5}}{\sqrt{\epsilon}}\right)$

- 😊 Achieve a better regret vs. $\tilde{O}(T^{3/4})$ under local model **without a trusted server**
- 😊 Minimal modification on existing private algorithms, i.e., batch + shuffler
- 😞 Privacy guarantee holds only for small $\epsilon \ll 1$
- 😞 Continuous privacy noise, difficulty on finite computers and even privacy leakage [Kairouz et al. 2021, Mironov et al. 2012]
- 😞 Communication of real numbers

SDP via Vector Sum

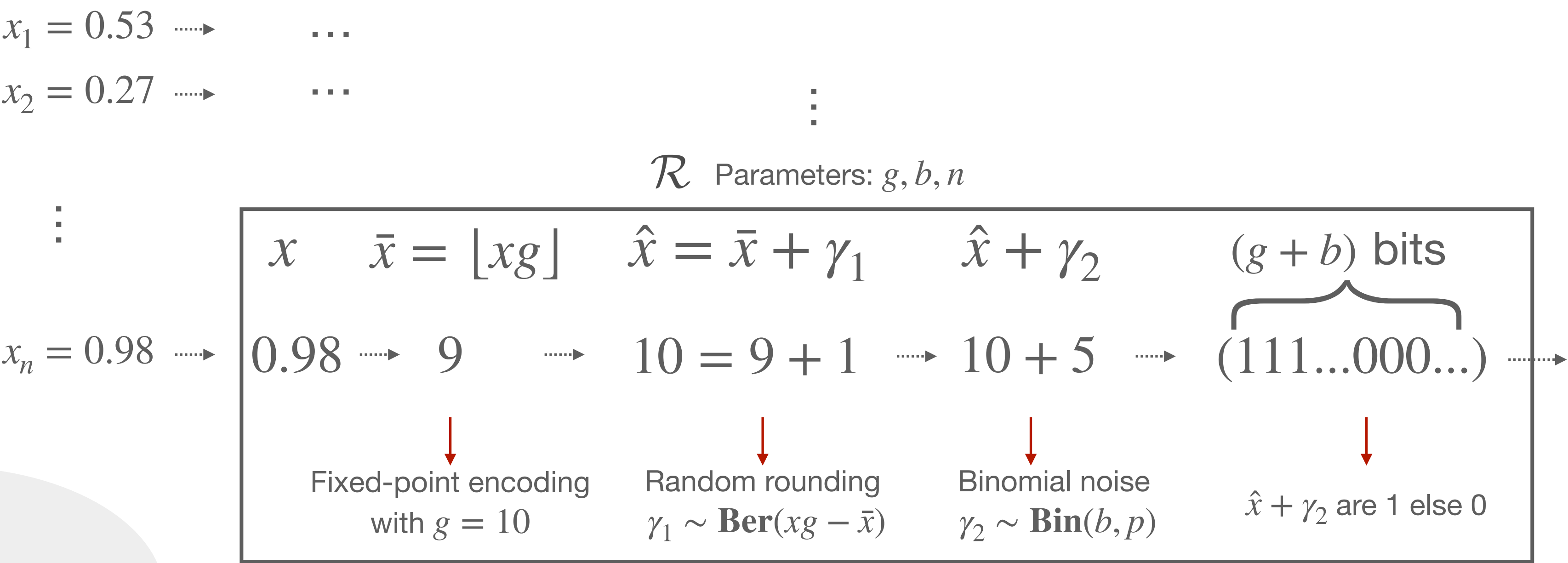
Shuffle Bounded Sum

Introduction

- **Problem.** Given n numbers within $[0,1]$, private sum with error $\tilde{O}(1/\epsilon)$, *no trusted server?*
- **A shuffle protocol.** $\mathcal{P} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ proposed in [Cheu et al. 2021 ✨]
 - Randomizer — fixed-point encoding + random rounding + Binomial noise
 - 😊 only discrete noise + bit communication
 - Shuffler — randomly permute a bunch of bits
 - Analyzer — aggregate bits with simple de-bias operation

Shuffle Bounded Sum

Illustration $\mathcal{P}_{1D} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$

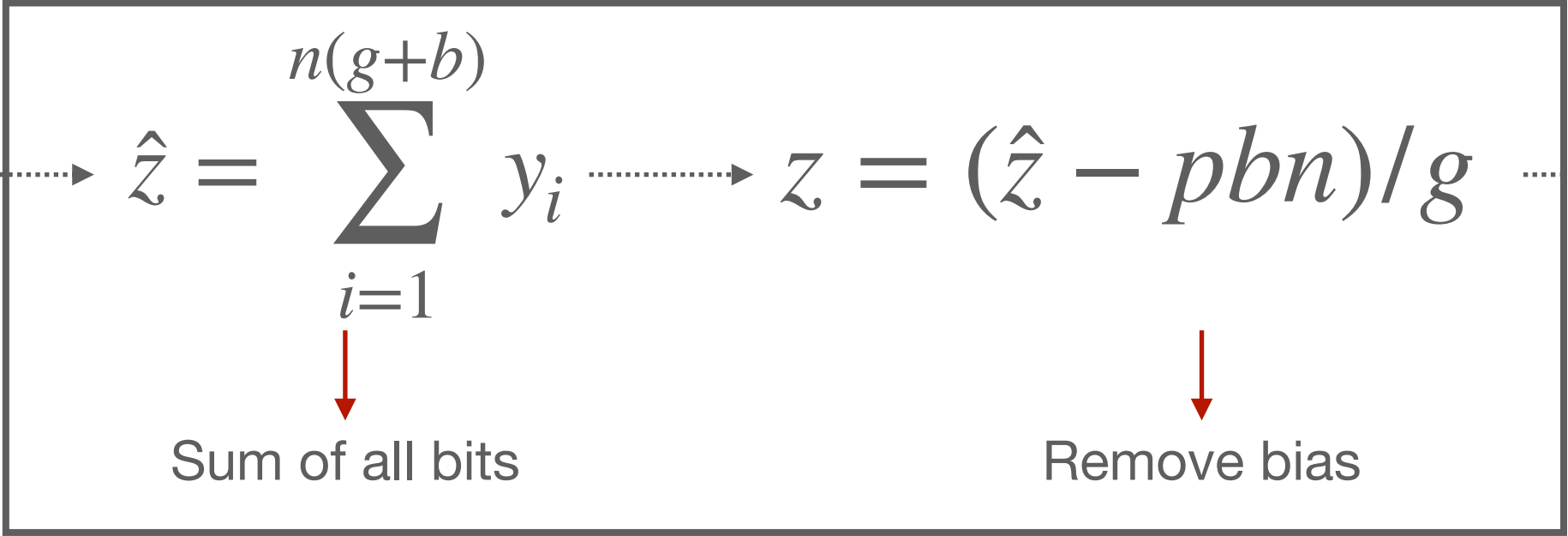


Is this private?



(110101...100...)

$n \cdot (g + b)$ bits



$$z \approx \sum_i x_i$$

How close is it?



Shuffle Bounded Sum

Privacy and utility [Cheu et al. 2021 ✨]

- **Sum of n real $[0,1]$ numbers.** Let $g \geq \sqrt{n}$, $b = \tilde{O}(g^2/(\epsilon^2 n))$, $p = 1/4$

$\mathcal{P}_{1D} = (\mathcal{R}, \mathcal{S}, \mathcal{A})$ is (ϵ, δ) -SDP and z is unbiased with variance $\tilde{O}(1/\epsilon^2)$

- **“Amplification” of Binomial mechanism.**

- Each user injects binomial noise with variance $\approx bp = O(g^2/(\epsilon^2 n))$ with sensitivity g

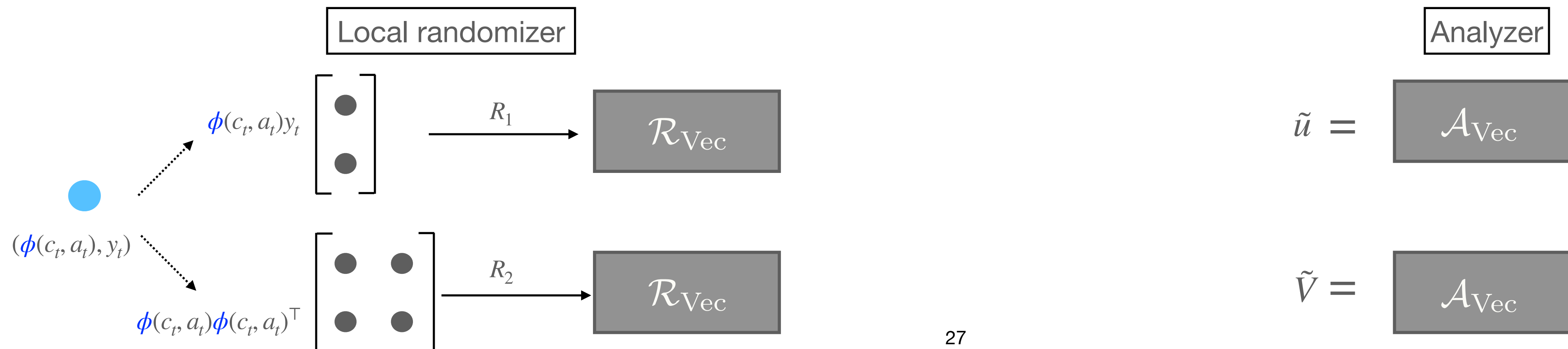
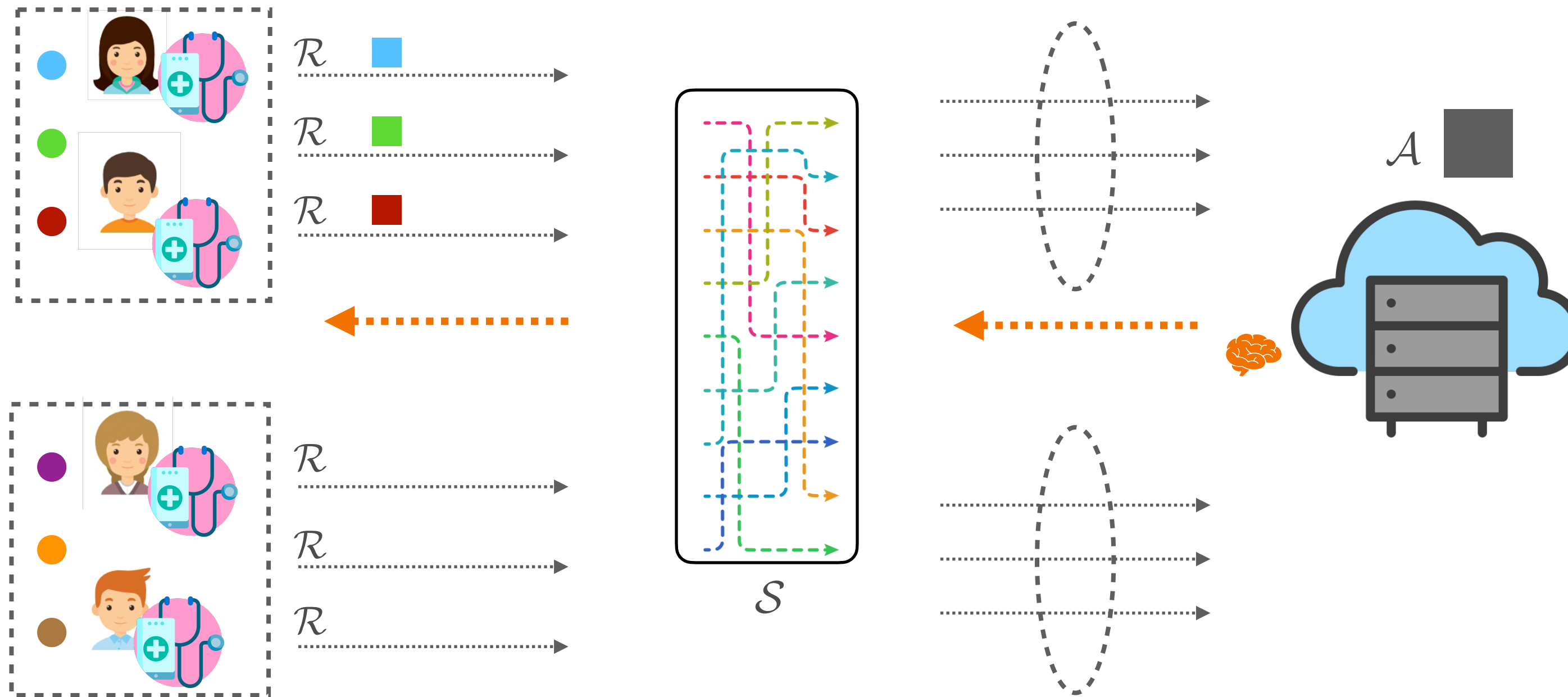
- Hence, it is $\epsilon_0 = \epsilon\sqrt{n}$ locally private by Binomial mechanism [Ghazi et al. 2019]

- **Sum of n norm-bounded vectors.** There exists parameters g, b, p , modification of \mathcal{P}_{1D}

coordinate-wise \mathcal{P}_{1D}
with additional shift

\mathcal{P}_{Vec} is (ϵ, δ) -SDP and the output of analyzer is unbiased with variance $\tilde{O}(d/\epsilon^2)$

Vector Sum in LCB



Performance

SDP via Vector Sum

Theorem

Fix batch size B , privacy budgets $\epsilon \in (0,15]$ and $\delta \in (0,1/2)$. There exist parameter choices of g, b, p , such that

- **(Privacy)** Our algorithm is (ϵ, δ) -SDP
- **(Regret)** Set $B = O(T^{3/5})$, with a high probability, our algorithm achieves $\tilde{O}\left(\frac{T^{3/5}}{\sqrt{\epsilon}}\right)$

- 😊 Achieve a better regret vs. $\tilde{O}(T^{3/4})$ under local model **without a trusted server**
- 😊 Privacy holds for $\epsilon > 1$
- 😊 Discrete noise and communicating bits
- 😞 Still has gap compared to central model $\tilde{O}(\sqrt{T})$

Proof Ideas

A Generic Regret Bound

◦ **Noise assumption.** Let n_i, N_i be total noised added in batch i for vector and matrix.

- For each m , $\sum_{i=1}^m n_i$ is a element-wise zero-mean sub-Gaussian with variance σ_1^2
- For each m , $\sum_{i=1}^m N_i$ is a element-wise zero-mean sub-Gaussian with variance σ_2^2
- Let $\sigma = \max\{\sigma_1, \sigma_2\}$

Lemma

Let above noise assumption holds. Our generic algorithm satisfies a high probability regret bound*

$$\text{Reg}(T) = \tilde{O} \left(dB + d\sqrt{T} + \sqrt{\sigma T} d^{3/4} \right)$$

Cost of batch update

Standard regret

Cost of privacy

A Generic Regret Bound

Applications

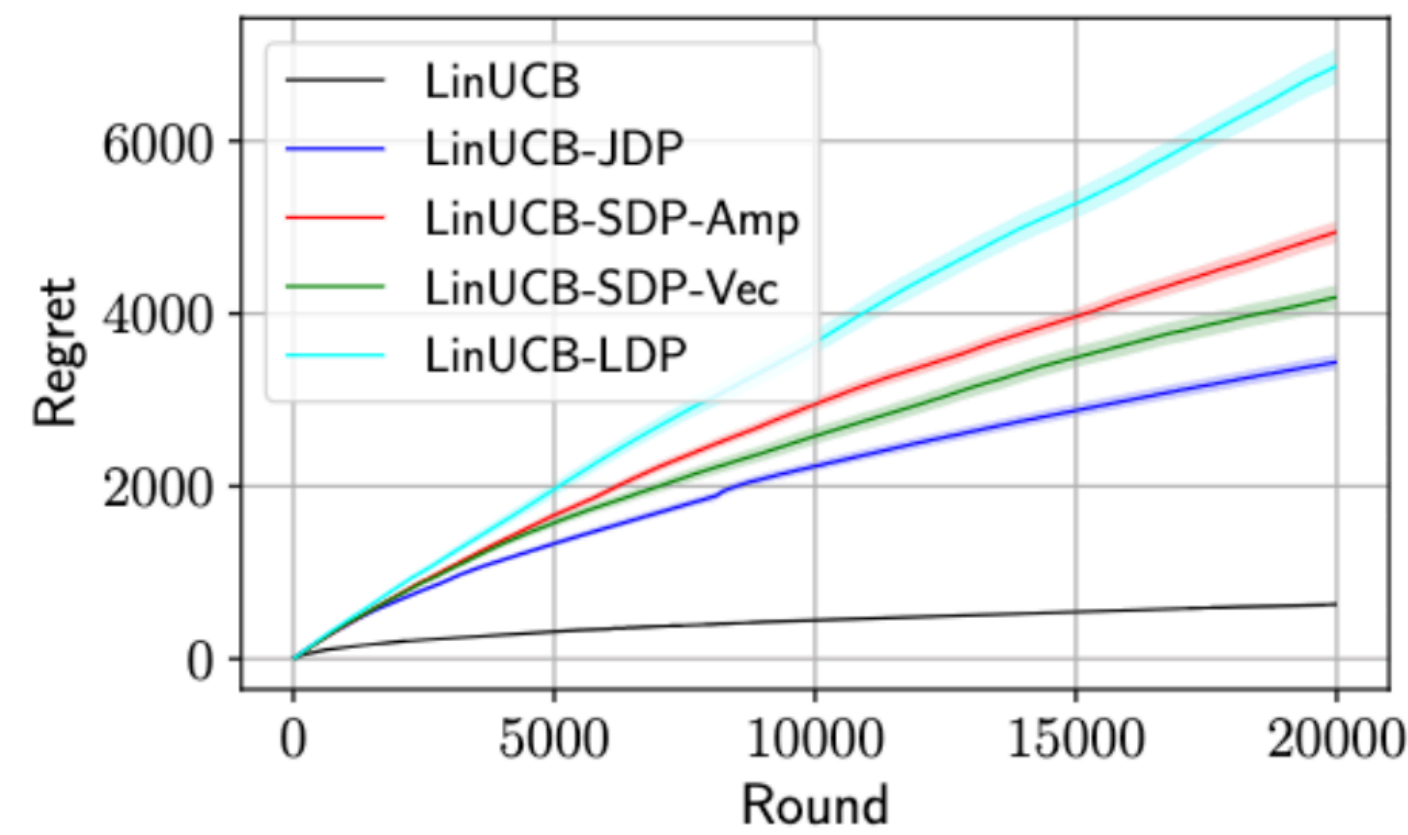
Lemma

Let noise assumption hold. Our generic algorithm satisfies a high probability regret bound

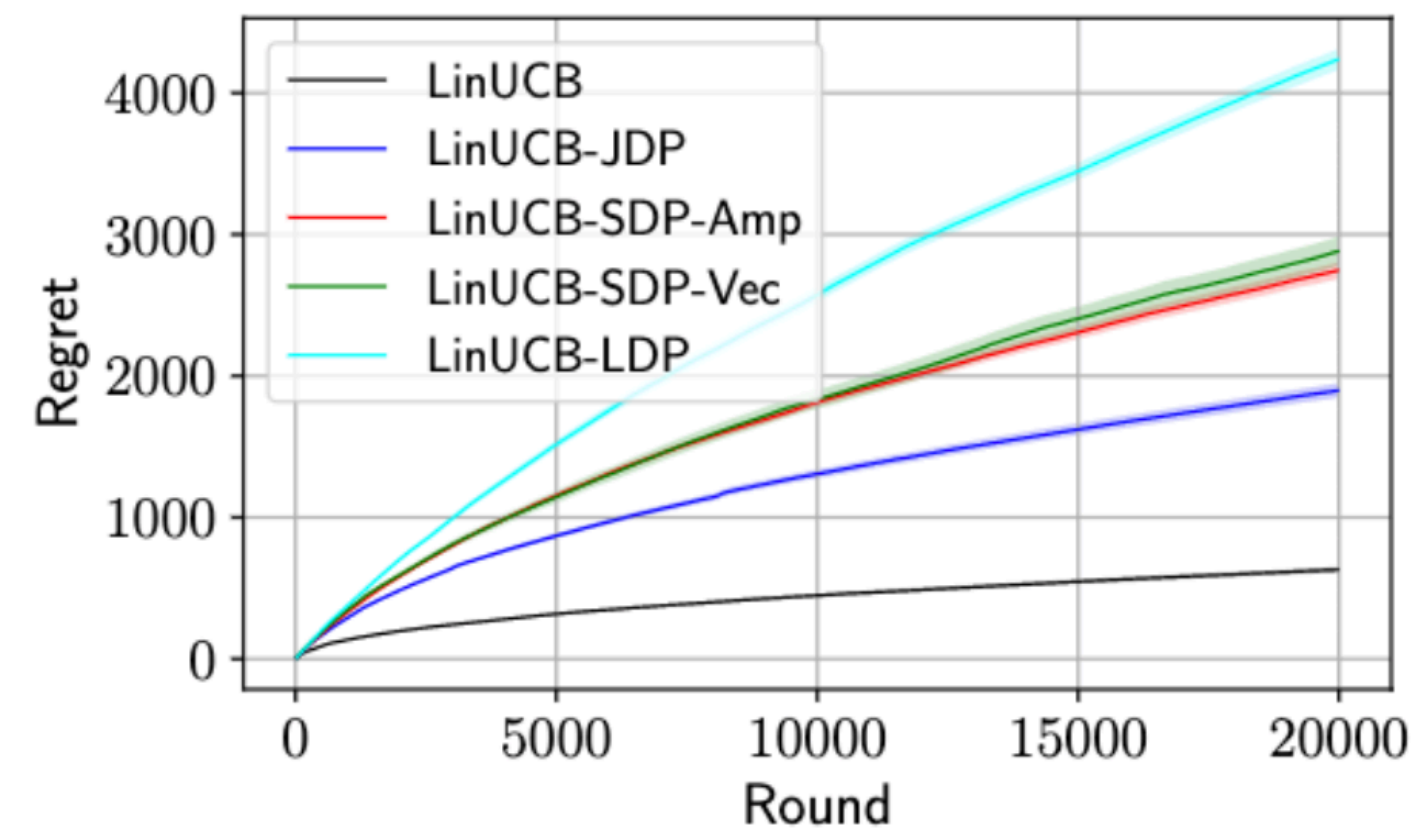
$$\text{Reg}(T) = \tilde{O} \left(dB + d\sqrt{T} + \sqrt{\sigma T} d^{3/4} \right)$$

- **SDP via LDP amplification** — $\sigma^2 \approx O(T/(\epsilon^2 B))$
 - Each user's noise is Gaussian with variance $\tilde{O}(1/(\epsilon^2 B))$ and a total of T such noise
- **SDP via Vector sum** — $\sigma^2 \approx O(T/(\epsilon^2 B))$
 - Each batch is sub-Gaussian noise with variance $\tilde{O}(1/\epsilon^2)$ and a total of $M = T/B$ such noise
- **Recover standard private bounds when $B = 1$** — Central model: $\sigma^2 \approx \log T/\epsilon^2$ and Local model: $\sigma^2 \approx T/\epsilon^2$
- **Batched central and local models ... improve non-private batch LinUCB...**

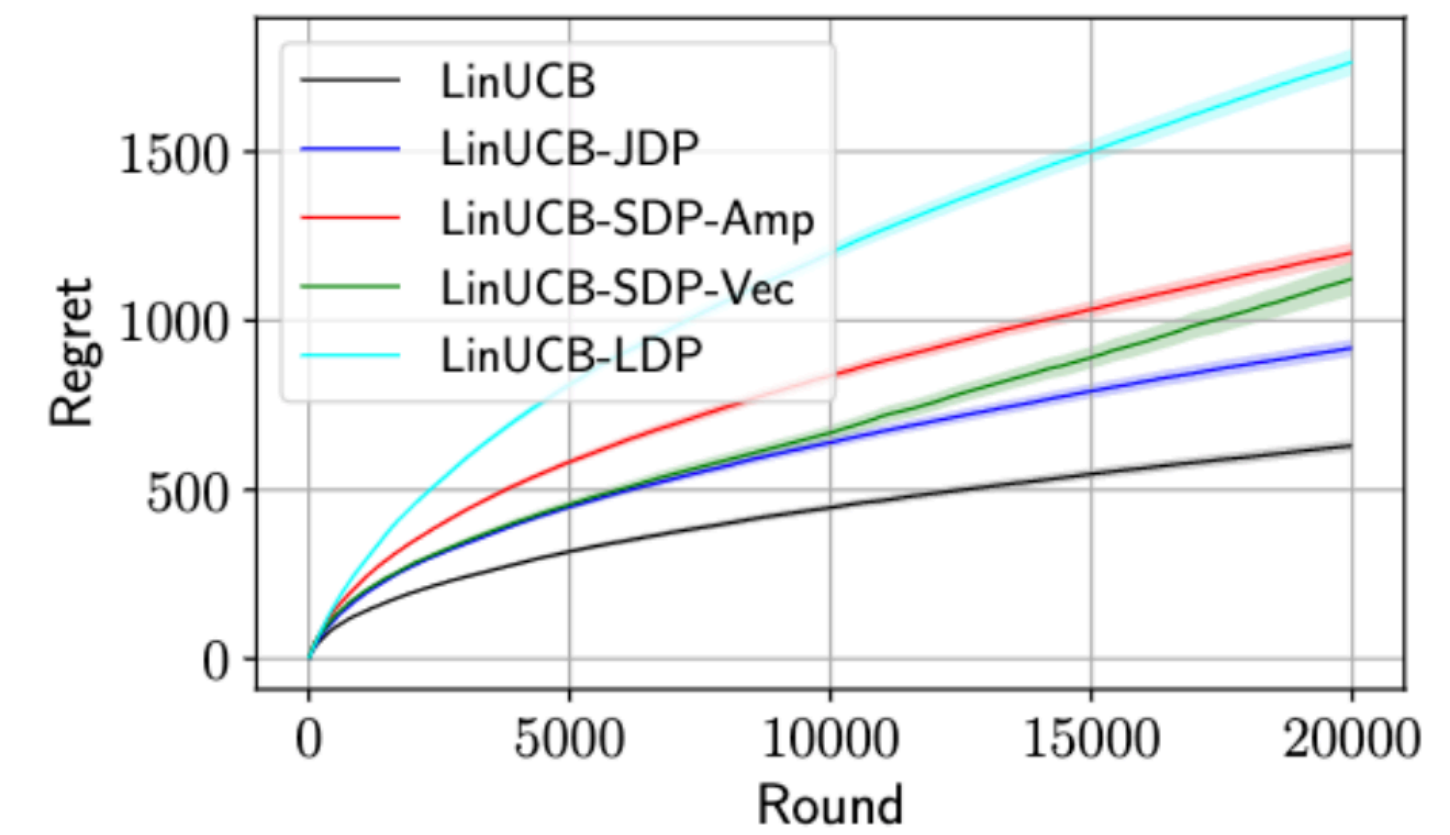
Simulations



(a) $\epsilon = 0.2$



(b) $\epsilon = 1$





(c) $\epsilon = 10$

Our algorithm with **both** protocols achieve regret that lies **in between** central and local model

Returning Users

Introduction

- **Assumption.** Each user can participate *once* in all M batches
 -  Each batch — each phase of medical experiment
 -  Send feedback once in each phase allows for tracking the overall effectiveness
- **Key differences.**
 - Shuffle model — advanced composition of privacy loss is required
 - Central model — total sensitivity becomes larger
 - For central model, we consider users can participate in any M_0 rounds

Returning Users

Guarantees

Lemma

Let noise assumption hold. Our generic algorithm satisfies a high probability regret bound

$$\text{Reg}(T) = \tilde{O} \left(d\textcolor{red}{T}/\textcolor{red}{M} + d\sqrt{T} + \sqrt{\sigma T} d^{3/4} \right)$$

- **Shuffle model** — scale ϵ by $1/\sqrt{M}$ for (ϵ, δ) -SDP
 - As a result, total noise changes from $\sigma^2 \approx O(M/\epsilon^2)$ to $\sigma^2 \approx O(\textcolor{red}{M}^2/\epsilon^2)$
- **Central model** — scale ϵ by $1/M_0$ for (ϵ, δ) -DP in the central model
 - As a result, total noise changes from $\sigma^2 \approx O(\log T/\epsilon^2)$ to $\sigma^2 \approx O(\textcolor{red}{M}_0^2 \log T/\epsilon^2)$

If $M = M_0 = T^{1/3}$, both models have the same regret $\tilde{O}(T^{2/3})$!

Discussion

Concurrent Work

[Garcelon. et al 2021]

- **A more complicated algorithm.**
 - Two different batch schedules: shuffler — fixed batch size; server — adaptive batch schedule
 - This is due to the fact that their analysis of single-batch schedule is **not tight**
 - Instead, our tighter analysis shows that single-batch schedule is sufficient for same regret
- **Privacy guarantees hold only for $\epsilon \ll 1$.**
 - Instead, our SDP via vector sum holds for $\epsilon > 1$
- **Adaptive batch schedule in fact causes trouble, i.e., a gap in Theorem 10 of their paper.**
 - The key issue is that standard determinant trick cannot be directly used
 - It relies on the fact that $V_t \geq V_{\tau_t}$, where $\tau_t < t$ is the most recent model update time
 - However, this does not necessarily hold due to the added privacy noise! (This problem exists for all three DP models)

Open Problems

- **Can we close the gap?**
 - What's the lower bound for local model? i.e., Can $O(T^{3/4})$ be improved?
 - Or, can one further improve $O(T^{3/5})$ in the shuffle model?
- **Can we achieve pure DP in all three models?**
 - The key challenge is a non-trivial matrix concentration bound with sub-exponential tails
- **Can we do adaptive batch schedule (i.e., rarely-switching) in private case?**
 - The key challenge is that standard determinant trick fails

Thank you!