On Differentially Private Federated Linear Contextual Bandits

Xingyu Zhou, Wayne State University

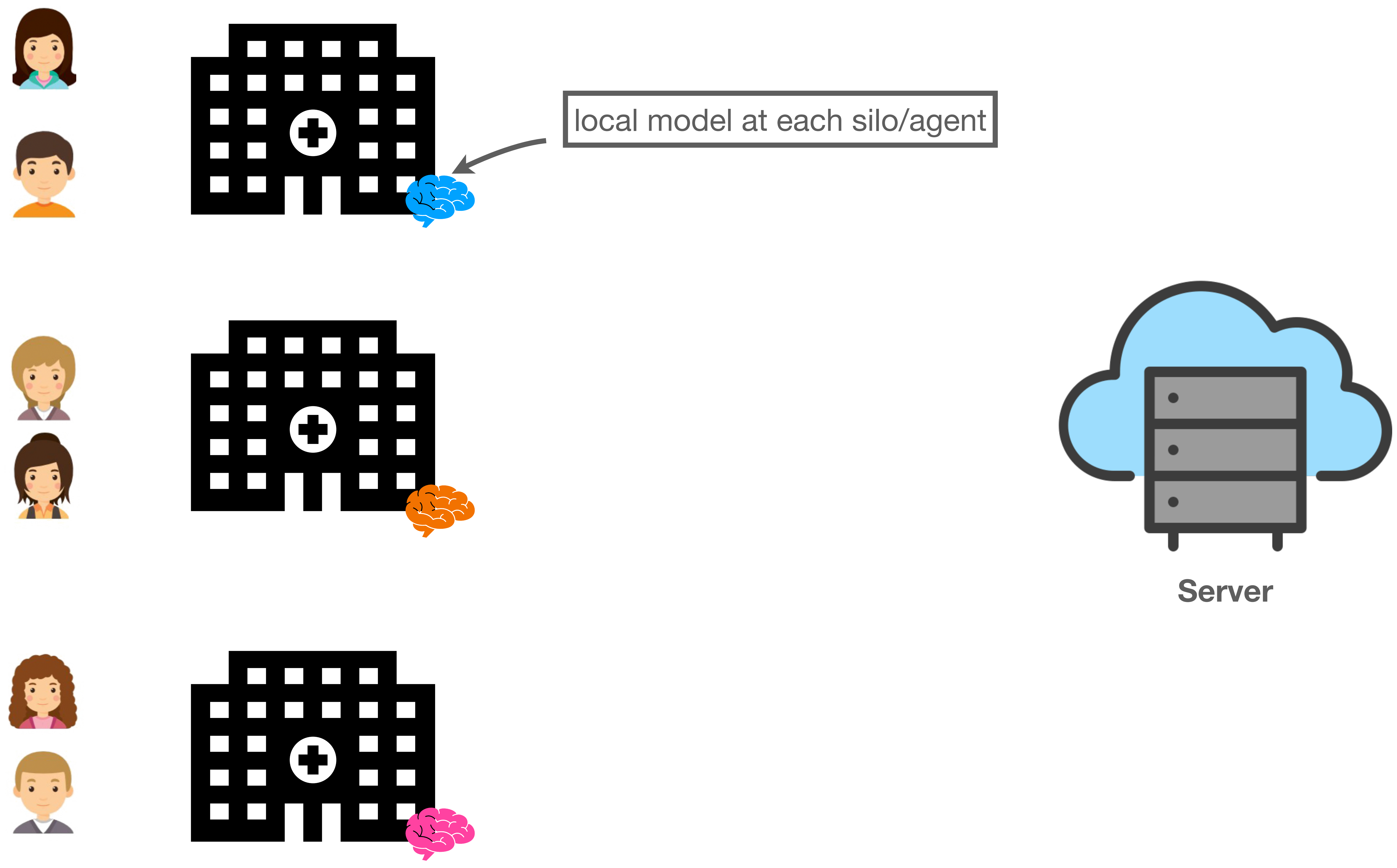
Oct. 18, 2023@ ASU

Hosted by Prof. Guoliang Xue

Joint work with **Sayak Ray Chowdhury**, Microsoft Research, India

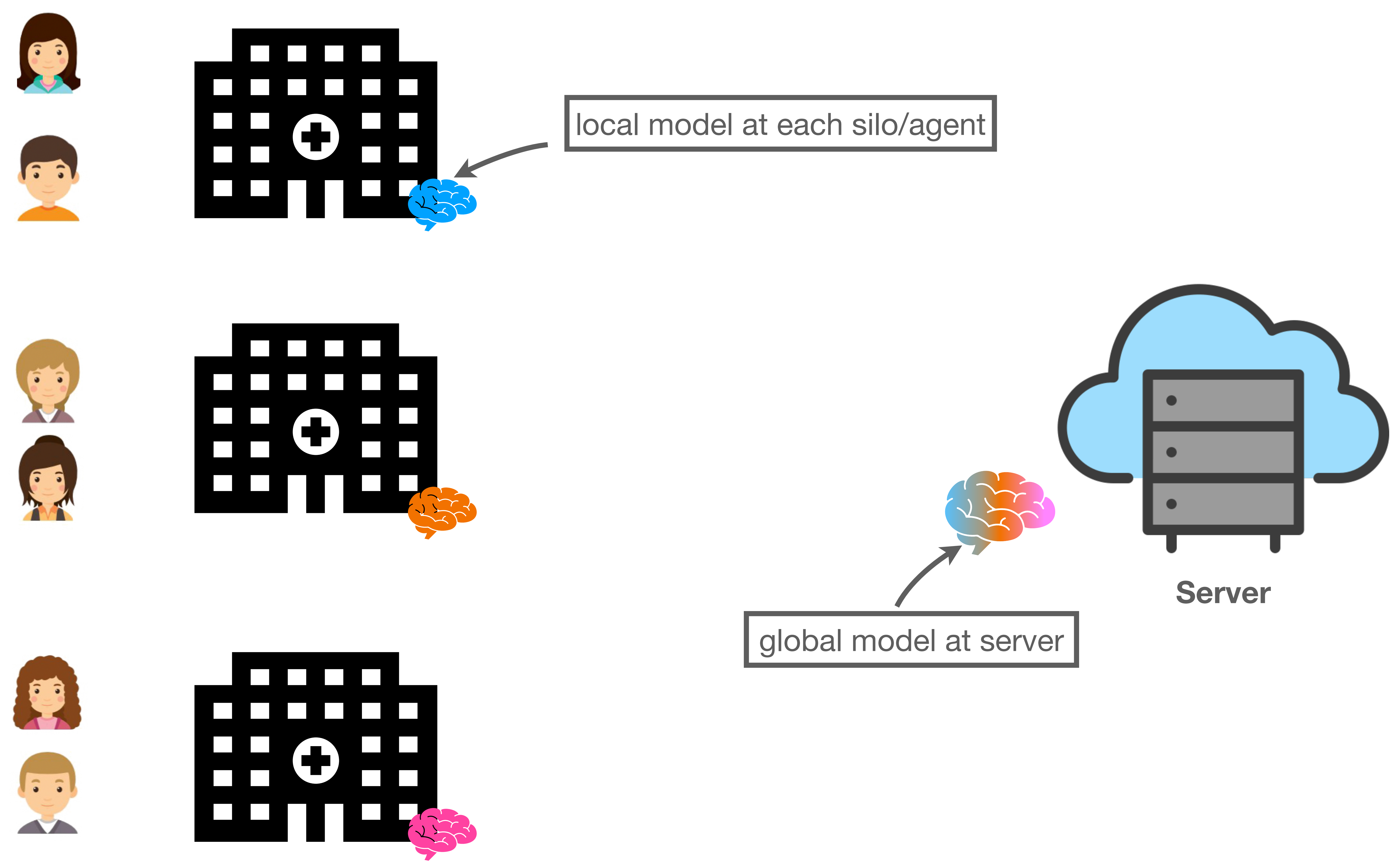
Cross-silo Federated Learning[KMA+19]

A hospital example



Cross-silo Federated Learning

A hospital example



Cross-silo Federated Learning

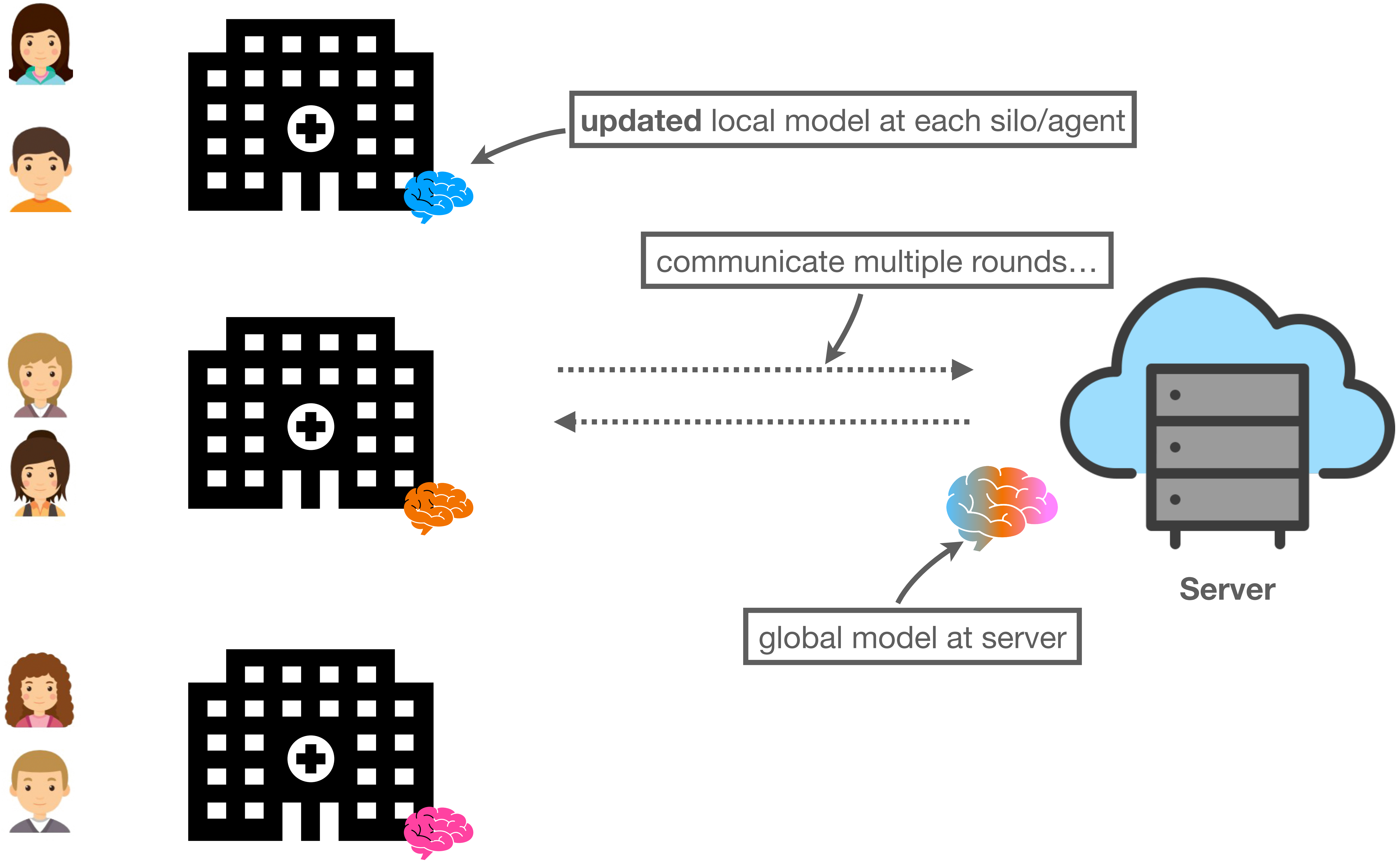
A hospital example

Cross-device FL

- Large no. of clients
- Limited resource
- e.g., clients are phones

Cross-silo FL

- Small no. agents/silos
- More resource
- e.g., silos are hospitals, banks, schools

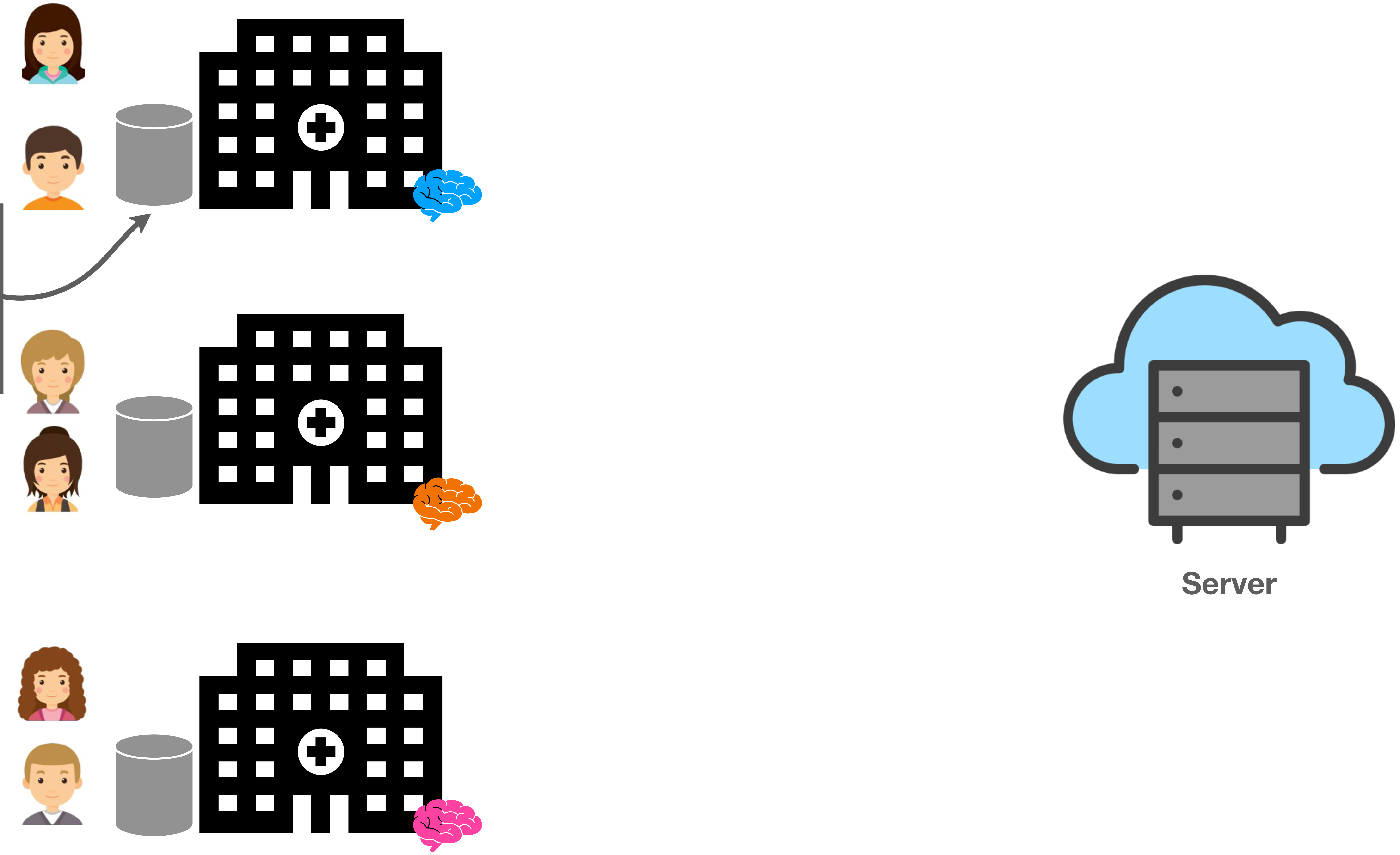


Cross-silo Federated Learning

Batch vs. Online learning

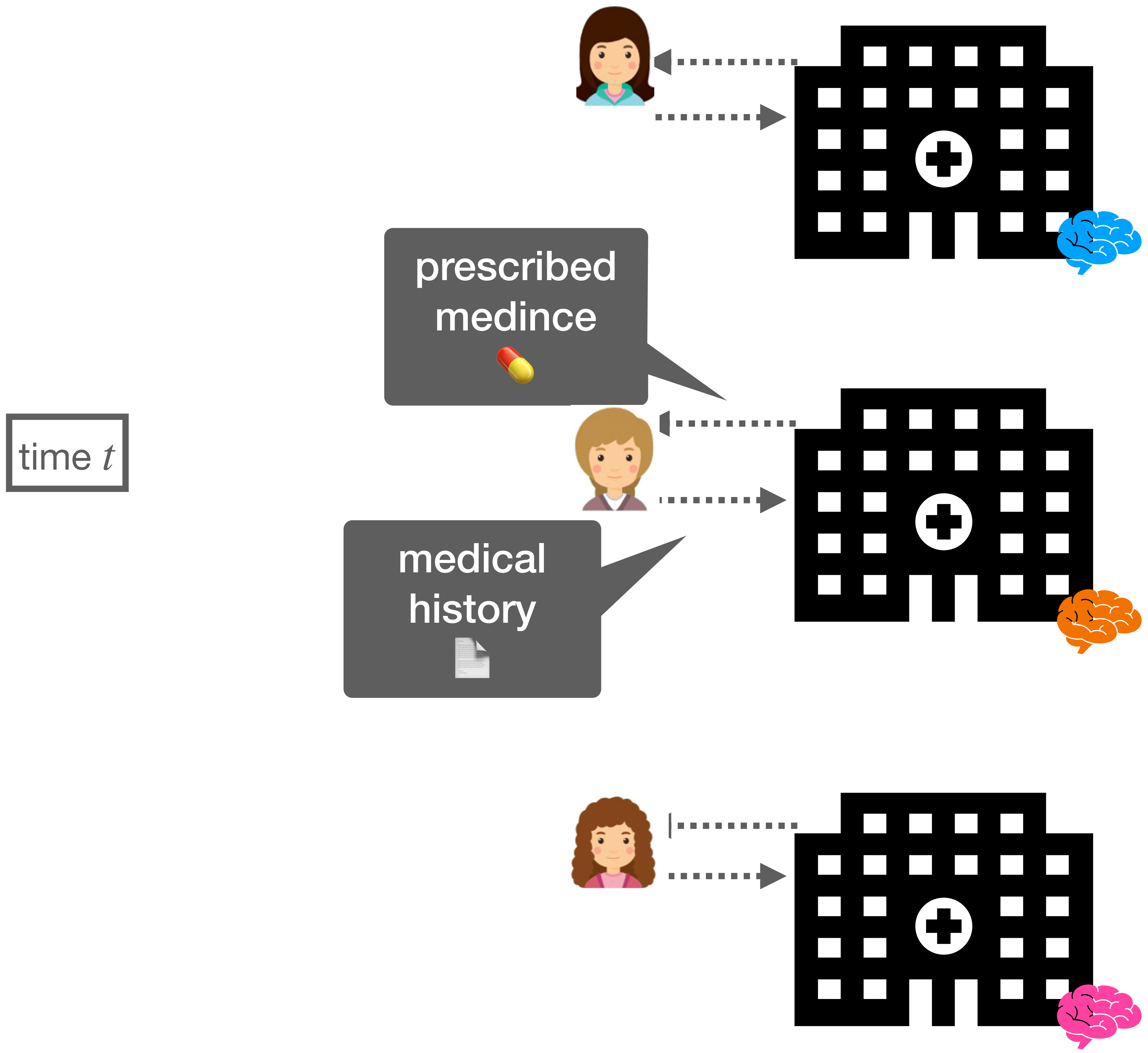
In FL **batch (supervised)** learning:

- each silo has a static dataset
- offline training
- e.g., cancer diagnosis



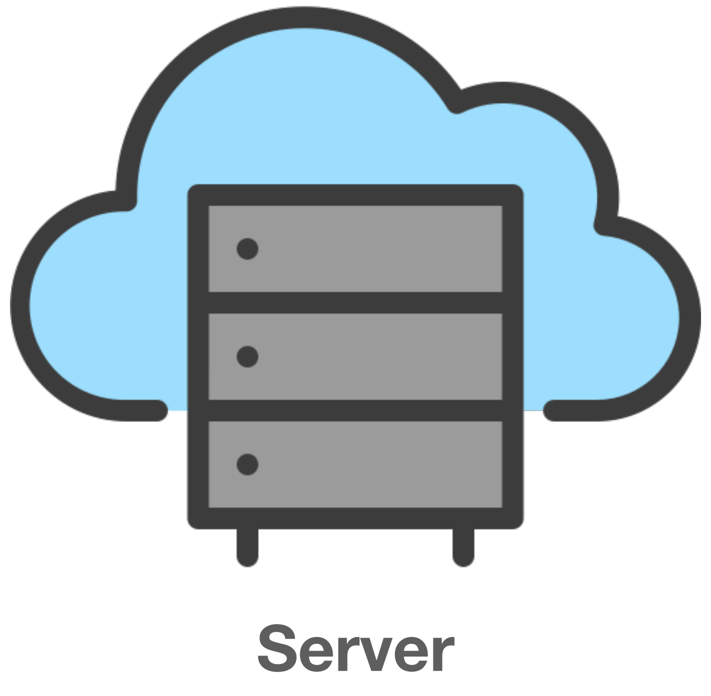
Cross-silo Federated Learning

Batch vs. Online learning



In FL **online** learning:

- each silo has a stream of data
- online training/decision, i.e., learn from interaction
- e.g., personalized medical care

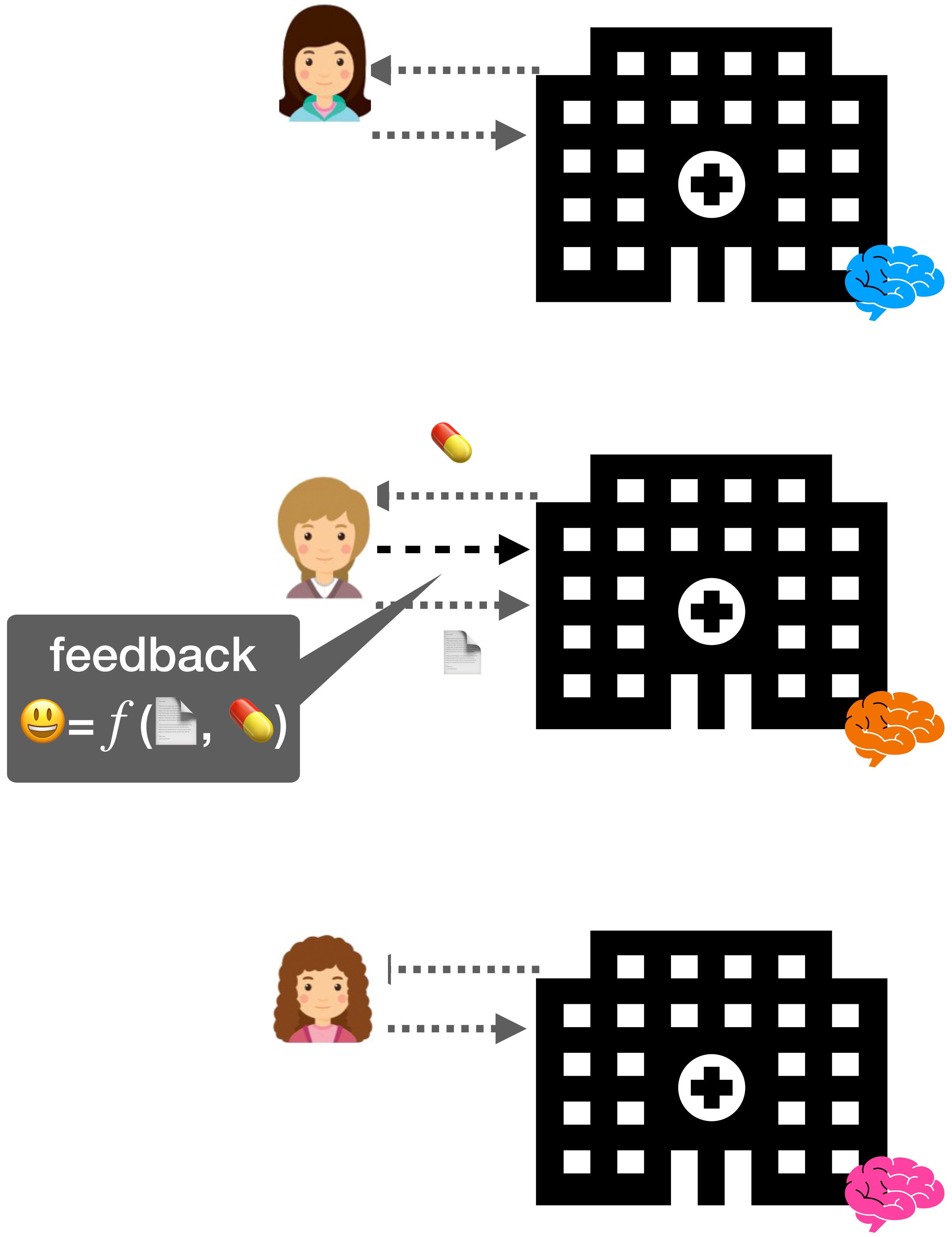


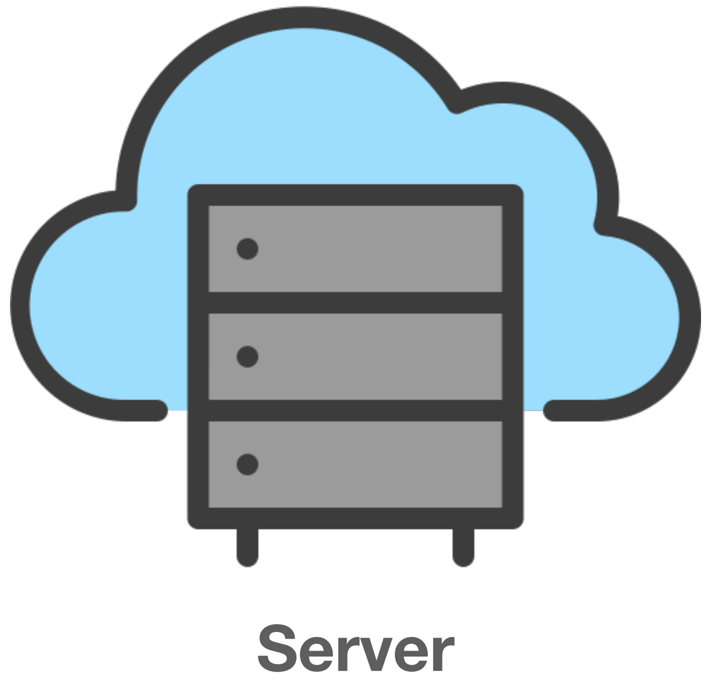
Cross-silo Federated Learning

Batch vs. Online learning

In FL **online** learning:
- each silo has a stream of data
- online training/decision, i.e., learn from interaction
- e.g., personalized medical care

time t

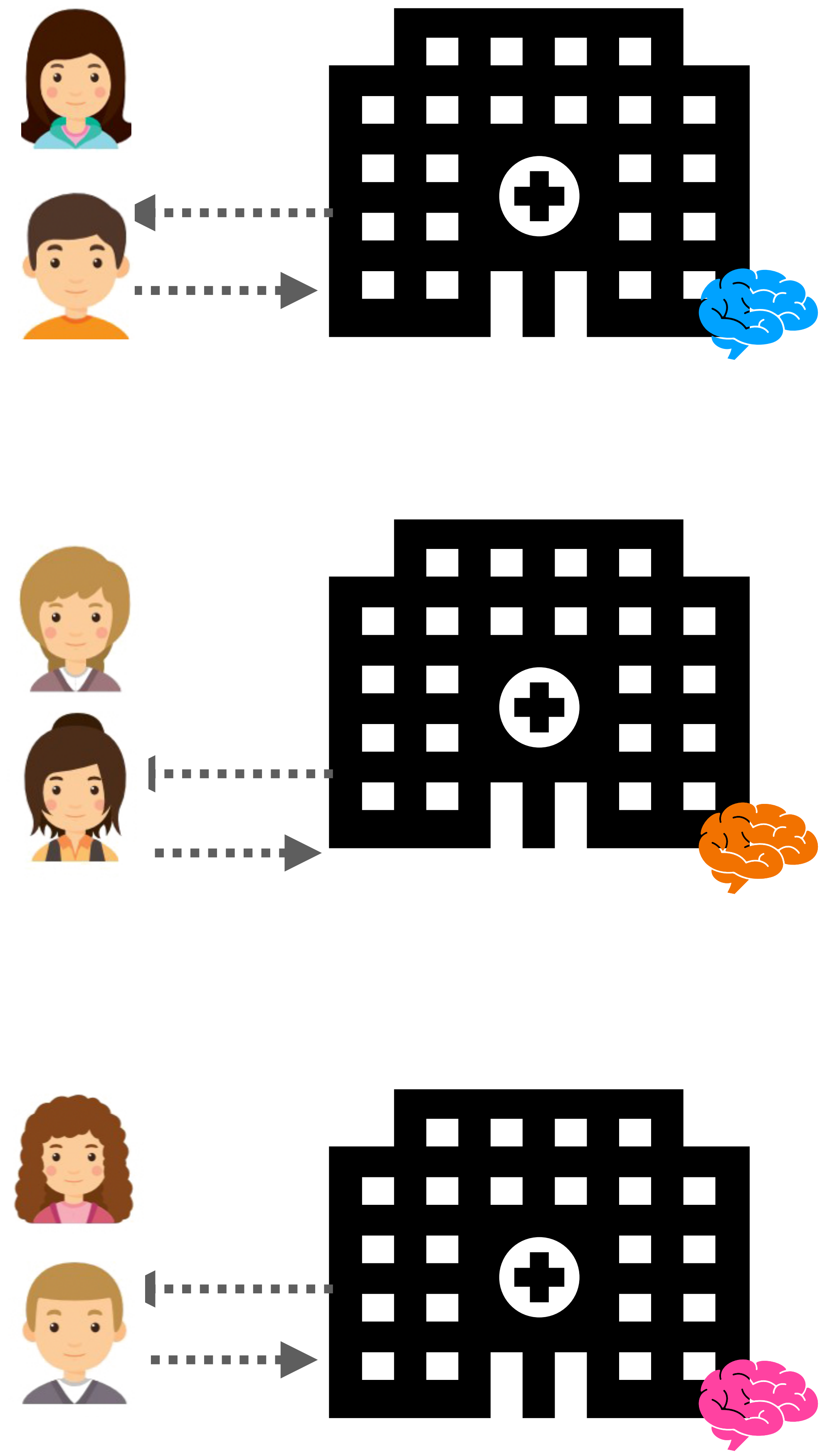




Cross-silo Federated Learning

Batch vs. Online learning

time $t + 1$

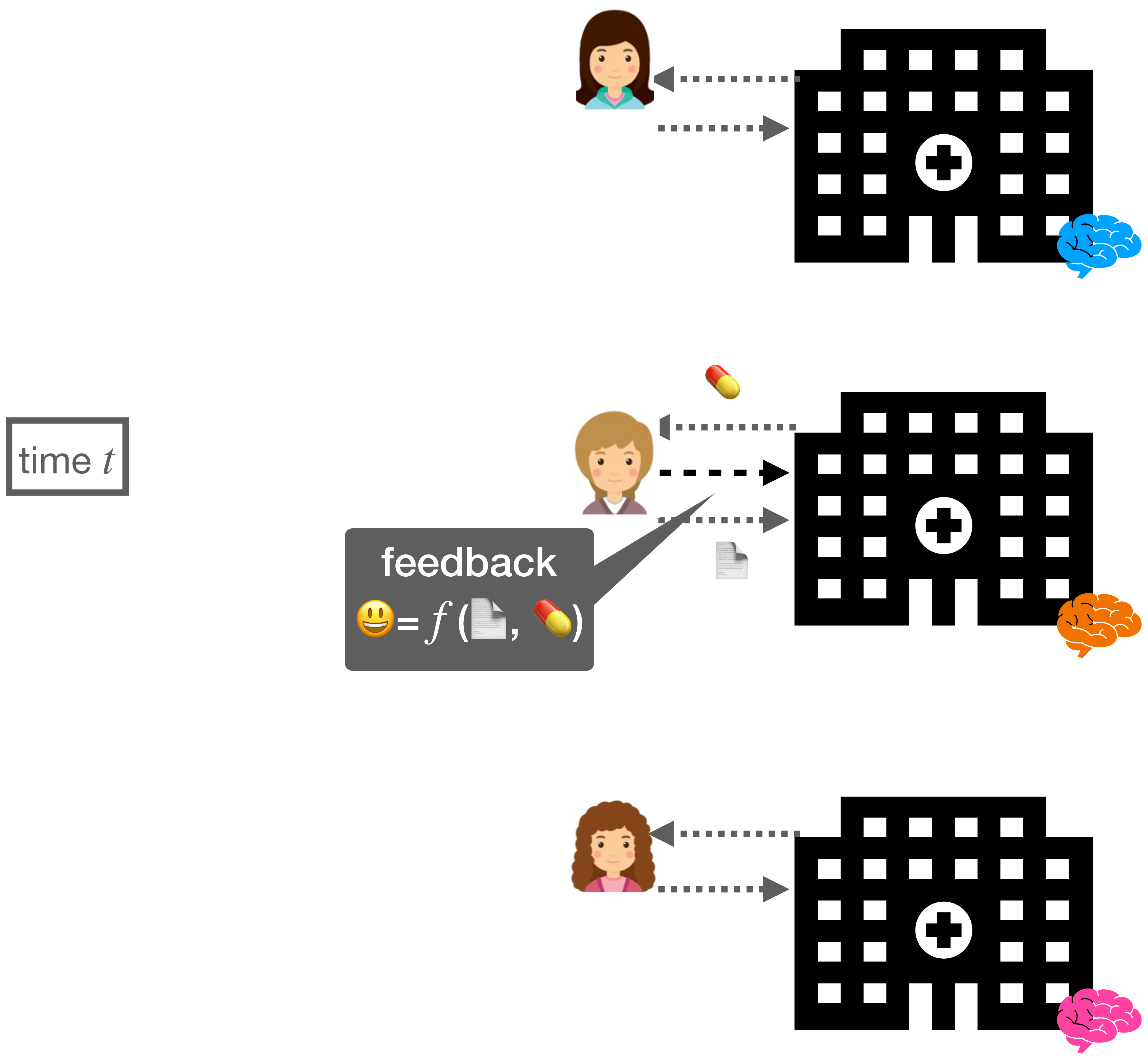


In FL **online** learning:

- each silo has a stream of data
- online training/decision, i.e., learn from interaction
- e.g., personalized medical care

Cross-silo Federated Learning

Linear contextual bandits (LCB)[APS11]



In FL **LCBs** (online):

- unknown reward feedback f is a **linear** function $y_t = x_t^\top \theta^* + \eta_t$
- $x_t = \phi(c_t, a_t) \in \mathbb{R}^d$, ϕ is the feature map, c_t is context and a_t is the action
- θ^* is the unknown parameter
- η_t is zero-mean noise

Server

Performance metric: **group regret** over M agents during T rounds

$$R_M(T) = \sum_{i=1}^{M} \sum_{t=1}^{T} \left[\max_a \phi_i(c_{t,i}, a)^\top \theta^* - \phi_i(c_{t,i}, a_{t,i})^\top \theta^* \right]$$

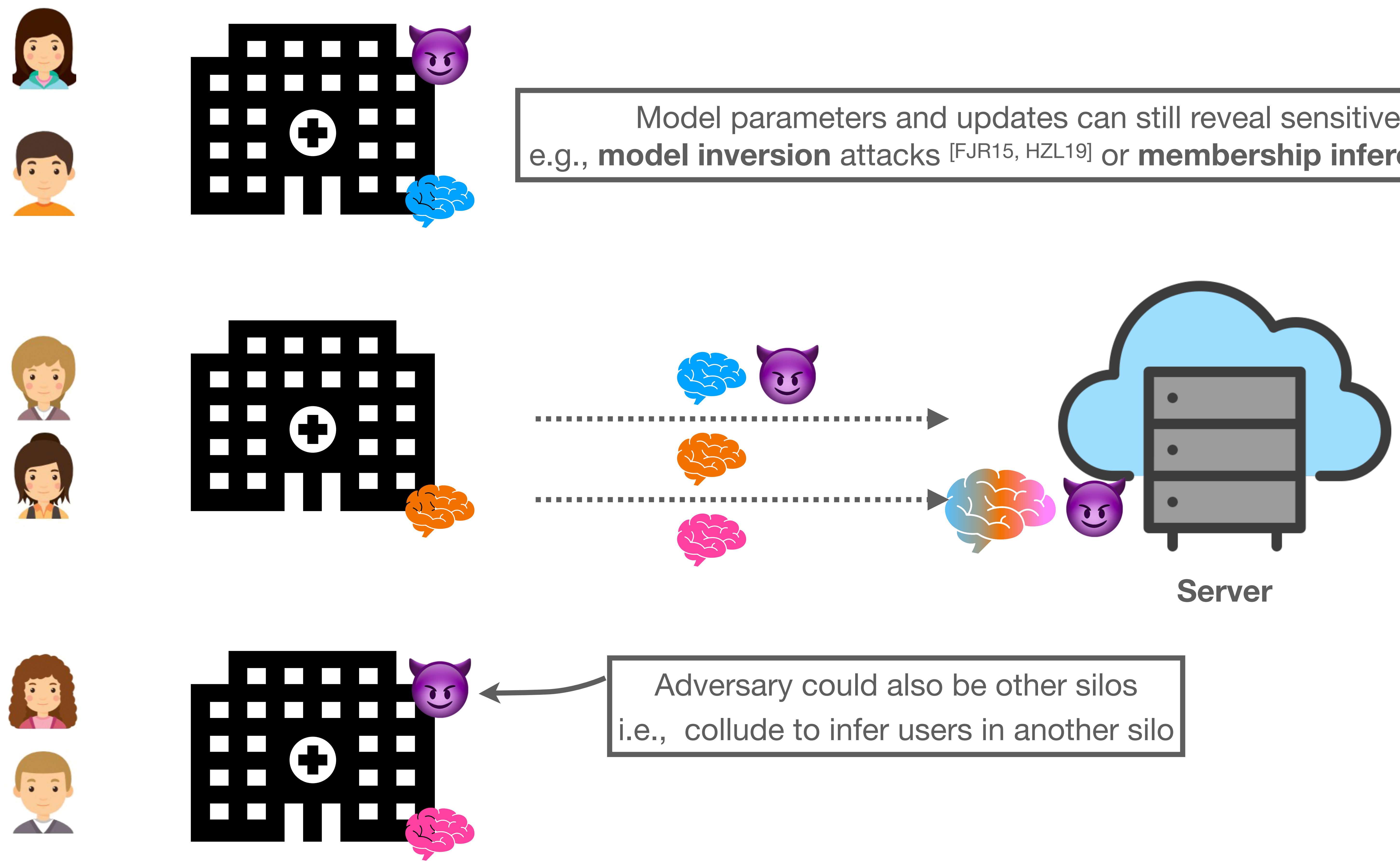
✅

On Differentially Private Federated Linear Contextual Bandits

✅

Privacy in Cross-silo FL

Though locally stored data, privacy risks still exist



Model parameters and updates can still reveal sensitive information
e.g., **model inversion** attacks [FJR15, HZL19] or **membership inference** attacks [SSS+17]

Adversary could also be other silos
i.e., collude to infer users in another silo

Differentially Private Cross-silo FL

Differential privacy[DR14] — a rigorous privacy protection

Differential Privacy 101

Definition. If for any two neighboring datasets D and D', and any outcome E

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

Then, M satisfies (ϵ, δ)-DP

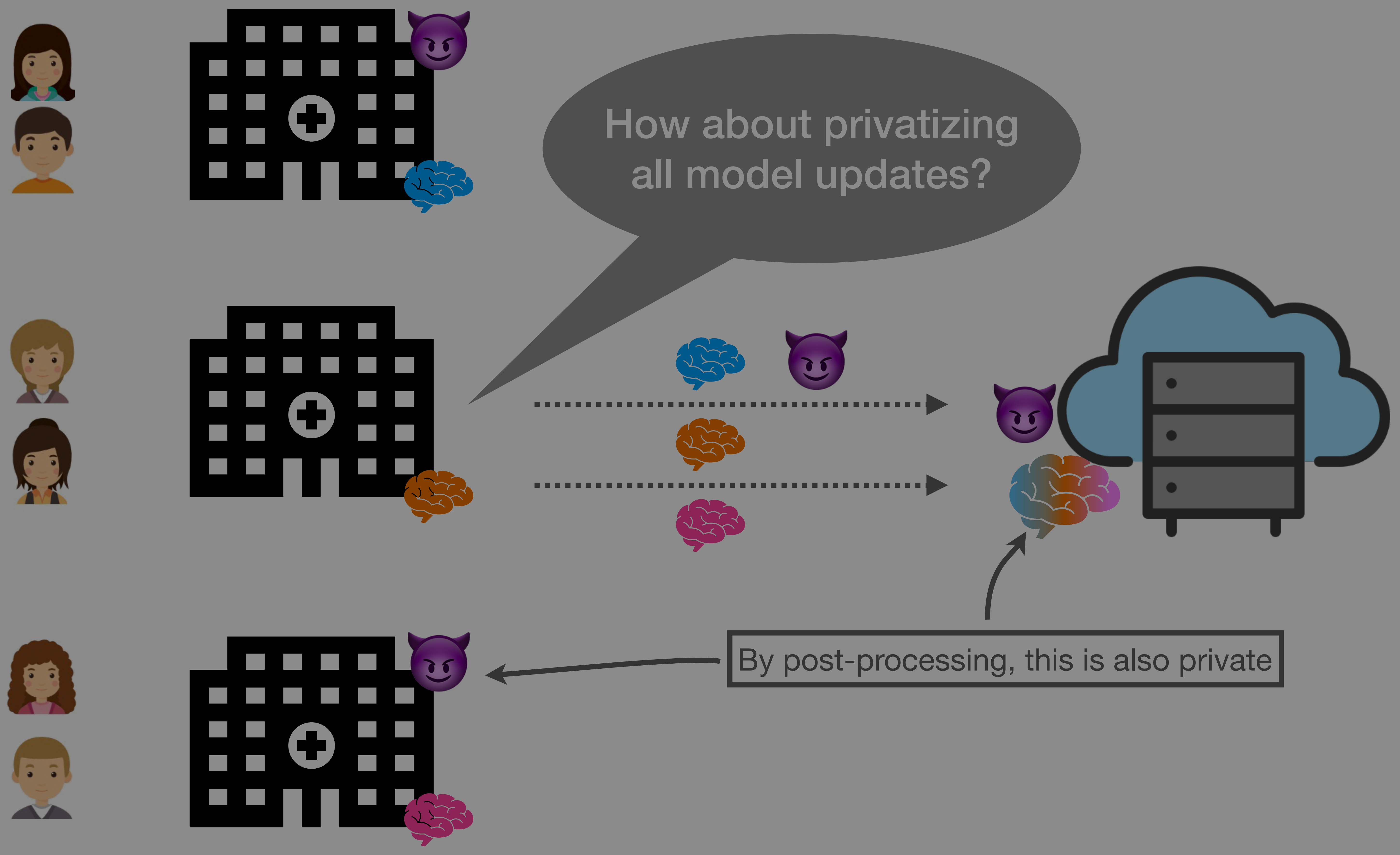
— DP means that outputs are "close" in probability[1] on two neighboring datasets

Key components:

1. *What are the neighboring datasets?*
   — the identity for protection
2. *What are the outputs?*
   — the view of adversary

Key properties:

1. *Composition*, privacy loss adds up
2. *Post-processing,* immune to further processing if data is not touched



[1] A more general view is via f-divergence or Renyi divergence between two distributions

Silo-level Local Differential Privacy (LDP)

All communication from each silo is private

Silo-level LDP [1]

Definition (informal). The full transcript of communication between any agent $i \in [M]$ and server are "close" in prob. on any two local neighboring datasets at agent i

Local neighboring datasets at agent i*: a sequence of* T *users that differs in only one user*

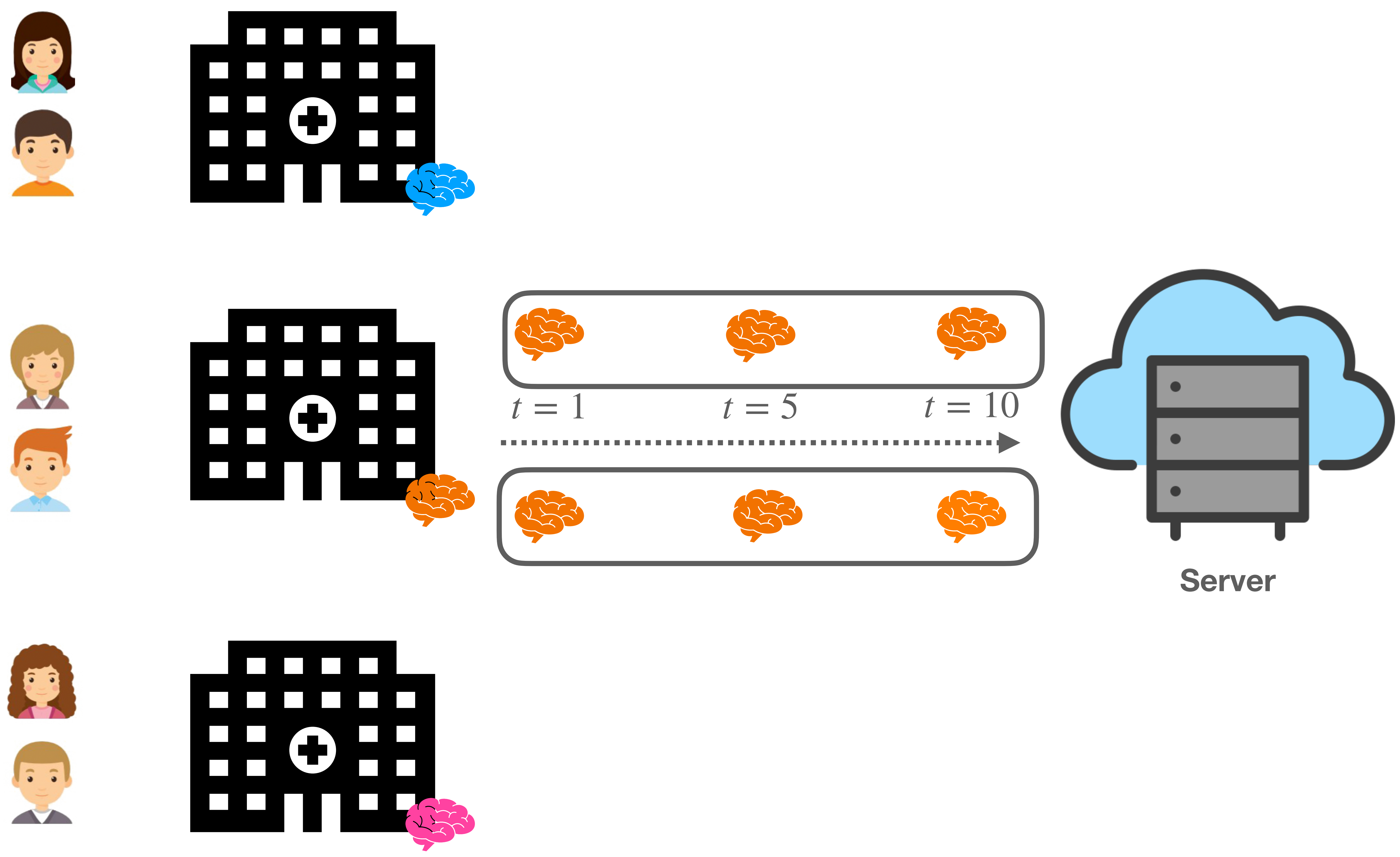
— protect each user/patient

— different from standard DP for cross-device FL, where each client is protected

Outputs: *full communication transcript*

— communicated models/messages

— communication schedule, i.e., when communication happens



[1] Similar privacy notion in different names exist in previous works, see [LR21, LHW+22]

On Differentially Private Federated Linear Contextual Bandits

✅ ✅ ✅

State-of-the-Art

Private Federated LCB

The state-of-the-art[DP20]

[Dubey&Pentland '20]

Algorithm: federated LinUCB with Gaussian mechanism (tree-based)

Privacy: essentially the same as silo-level LDP

Regret: *additional* regret due to privacy is $\tilde{O}(\sqrt{MT/\epsilon})$

Communication rounds: $O(\log T)$

Conclusion: match the regret achieved by a "super" single agent

Fundamental Issues

Privacy leakage

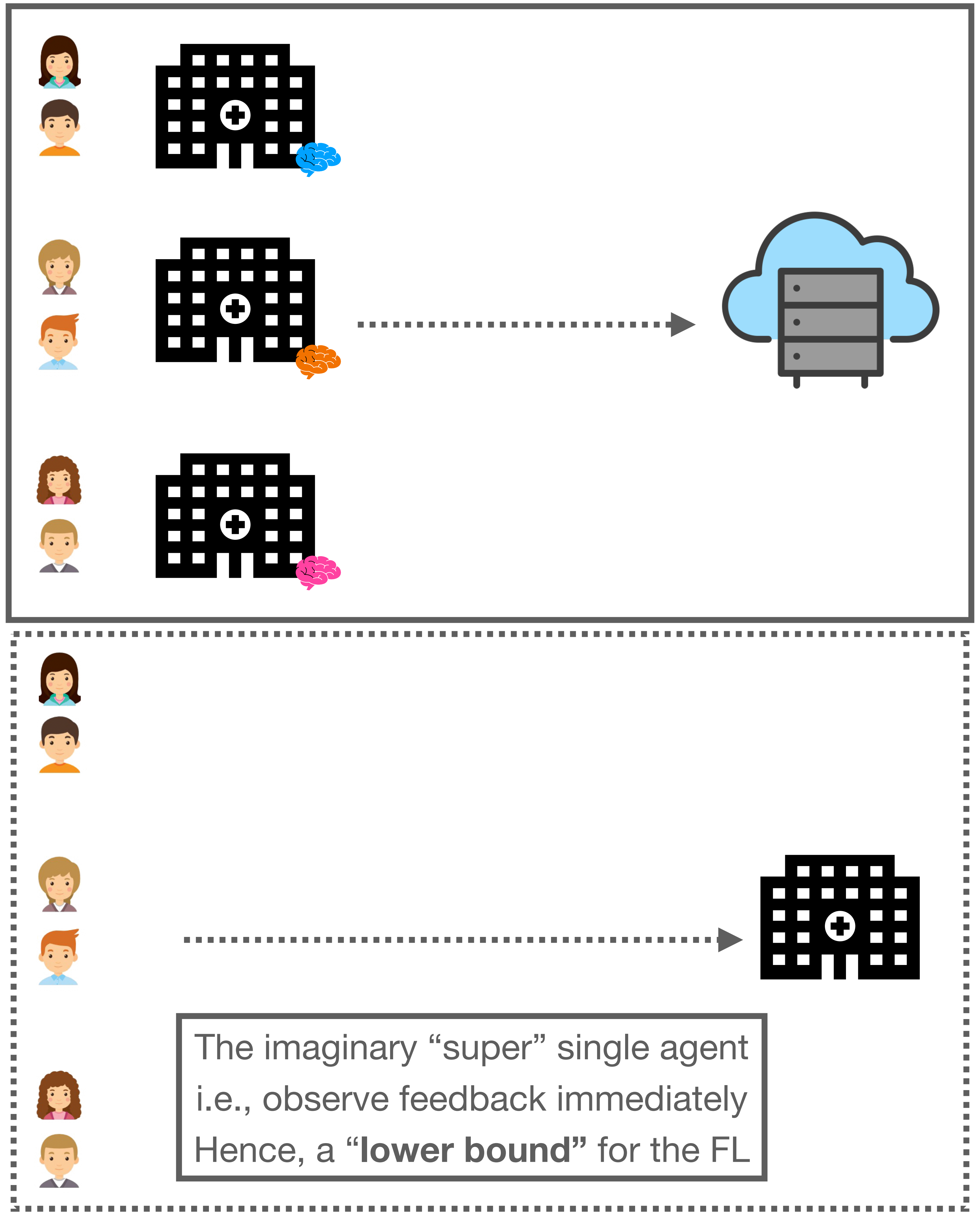
- The proposed algorithm fails to guarantee silo-level LDP
- A simple attack can reveal sensitive information of users

Incorrect regret

- The claimed privacy cost is mis-calculated
- The correct one is $\tilde{O}(M^{3/4}\sqrt{T/\epsilon})$
- Hence, no longer match the "lower bound"

Ungrounded communication cost

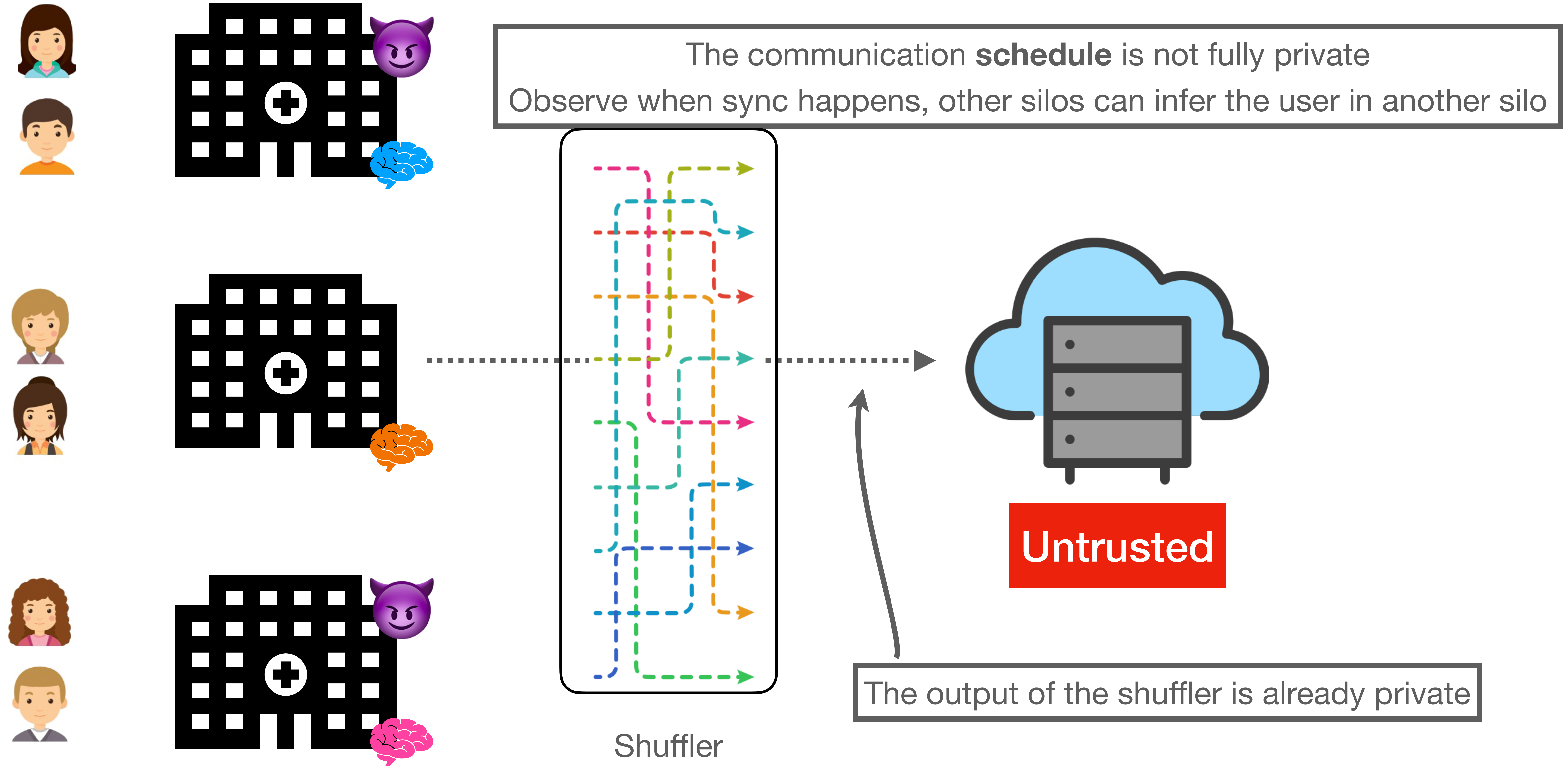
- The claimed $O(\log T)$ rounds comm. cost is not grounded



[DP20] Differentially-private federated linear bandits. *NeurIPS* 2020

Contribution

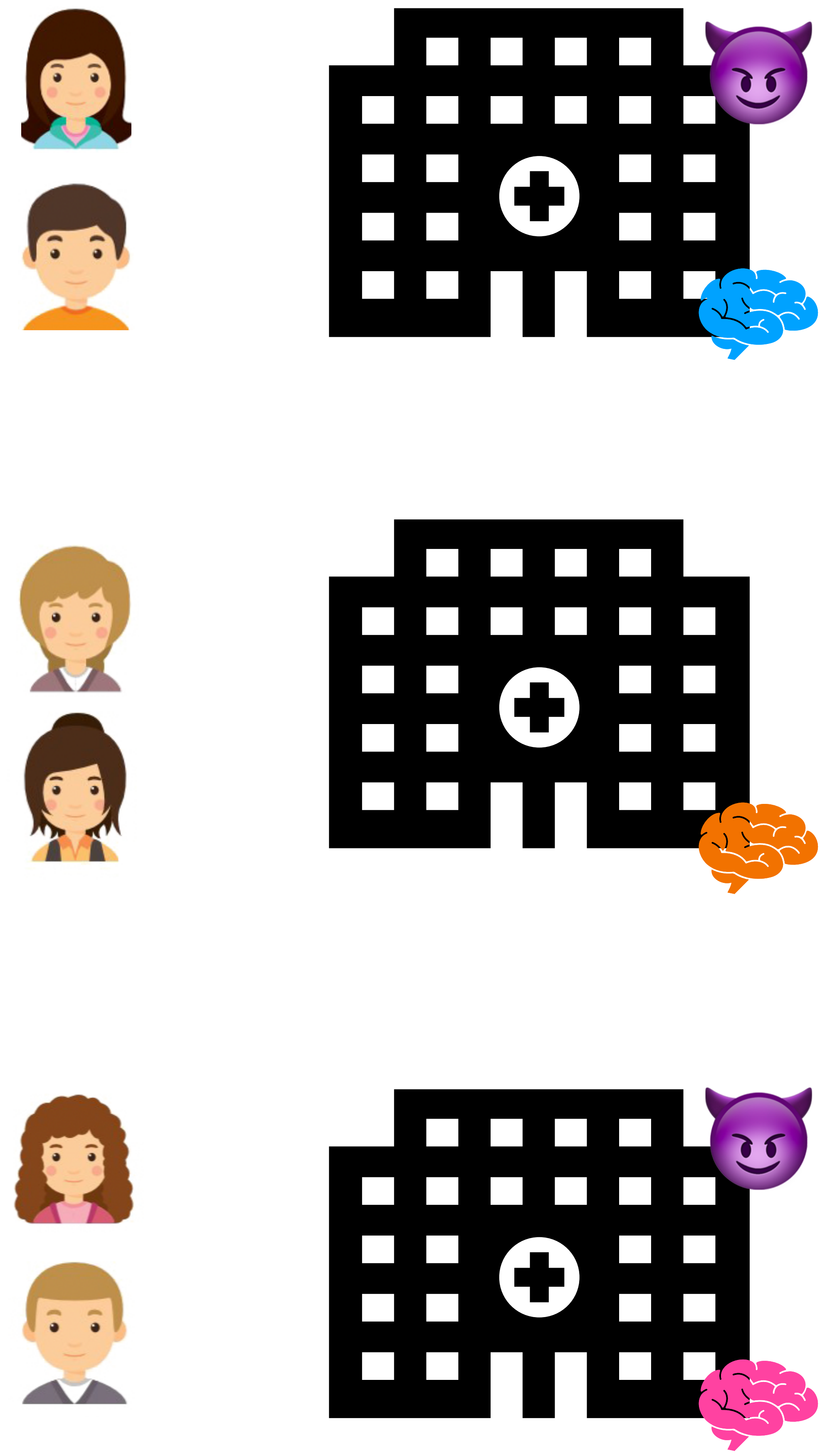
Main Results



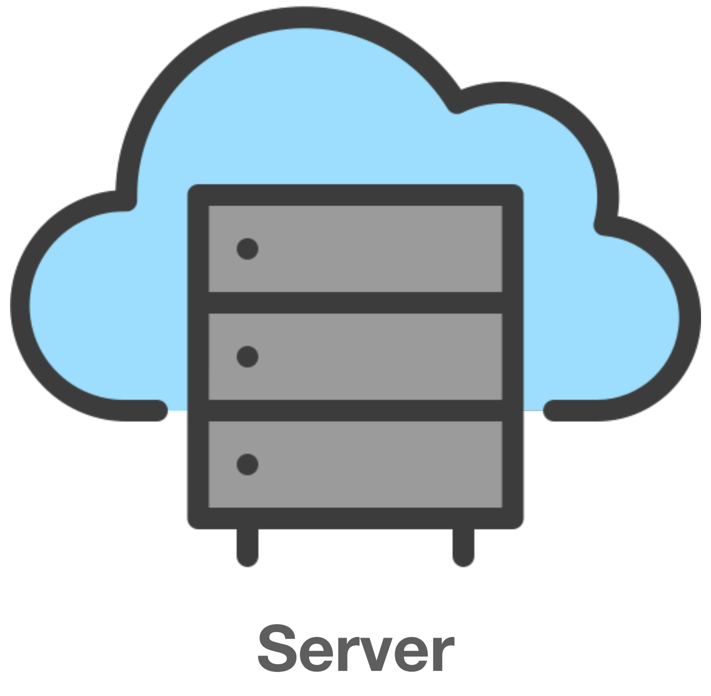
1. Identify the privacy, regret and communication cost gaps in the state-of-the-art
2. Propose a generic federated algorithm with flexible privacy protocols
3. Achieve the first correct regret bound under **silo-level LDP**, i.e., the privacy cost is $\tilde{O}(M^{3/4}\sqrt{T/\epsilon})$
4. Shave the additional $M^{1/4}$ factor under **shuffle differential privacy (SDP)** — still a strong privacy model

Privacy Gap in SOTA

Dynamic communication leaks privacy

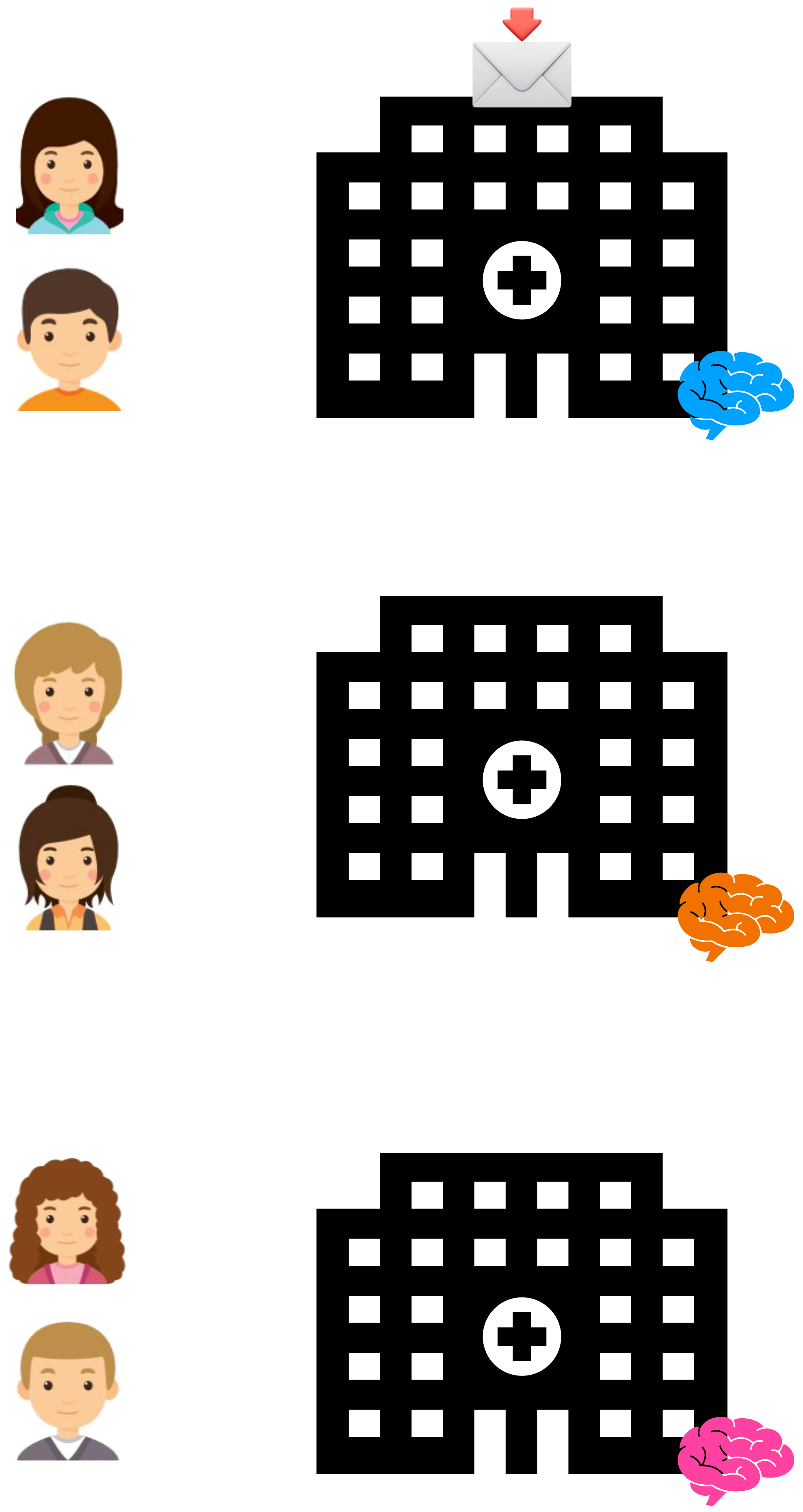


The communication **schedule** is not fully private
Observe when sync happens, other silos can infer the user in another silo



Privacy Gap in SOTA

Dynamic communication leaks privacy



Communication schedule for silos in SOTA

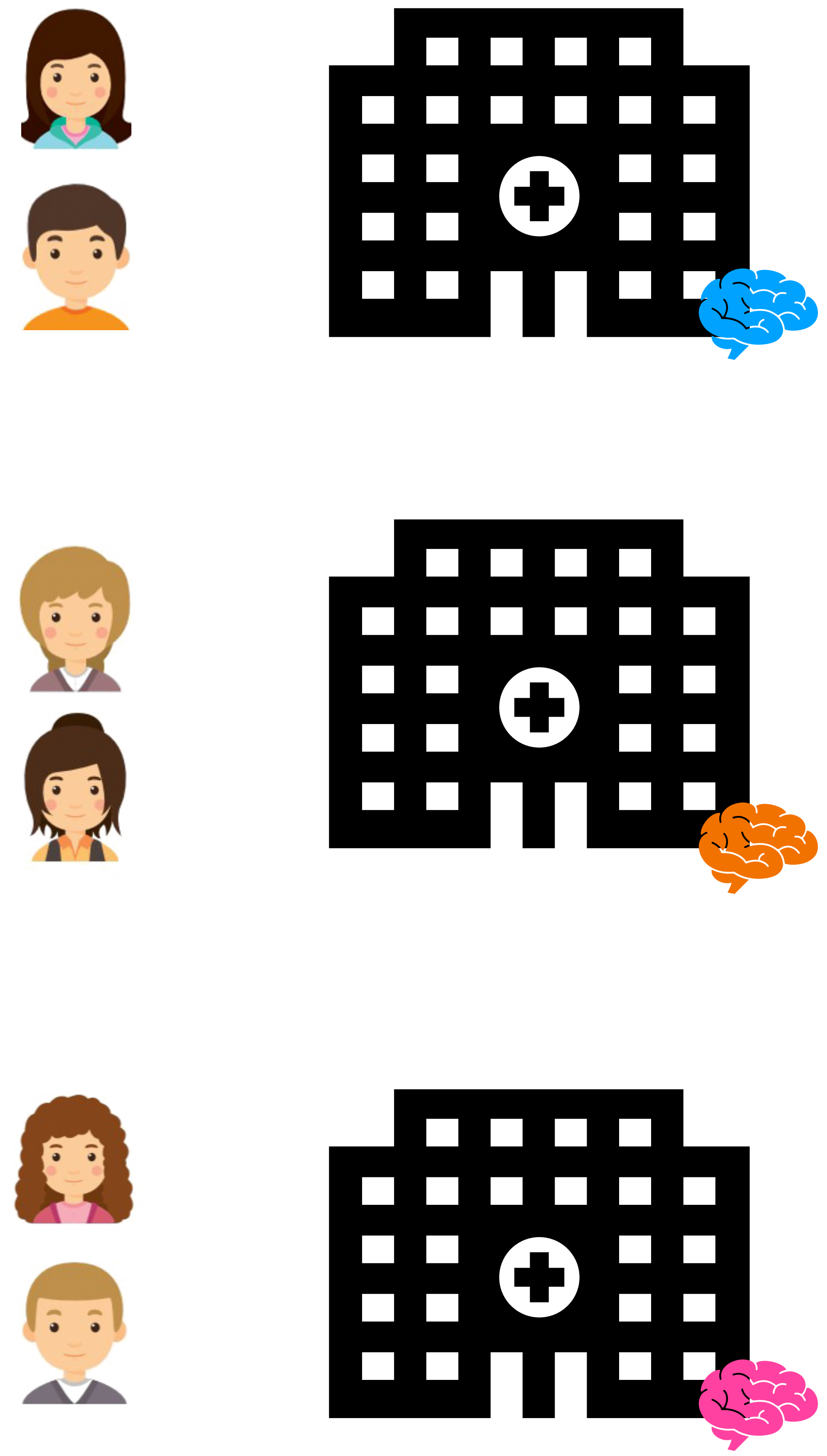
$$\exists i \in [M], \quad f(X_i, Z) > 0$$

- Z — all previous sync data among all silos
- X_i — **new non-private** local data at silo i since recent sync
- f — sync function, shared among all silos

Server

Privacy Gap in SOTA

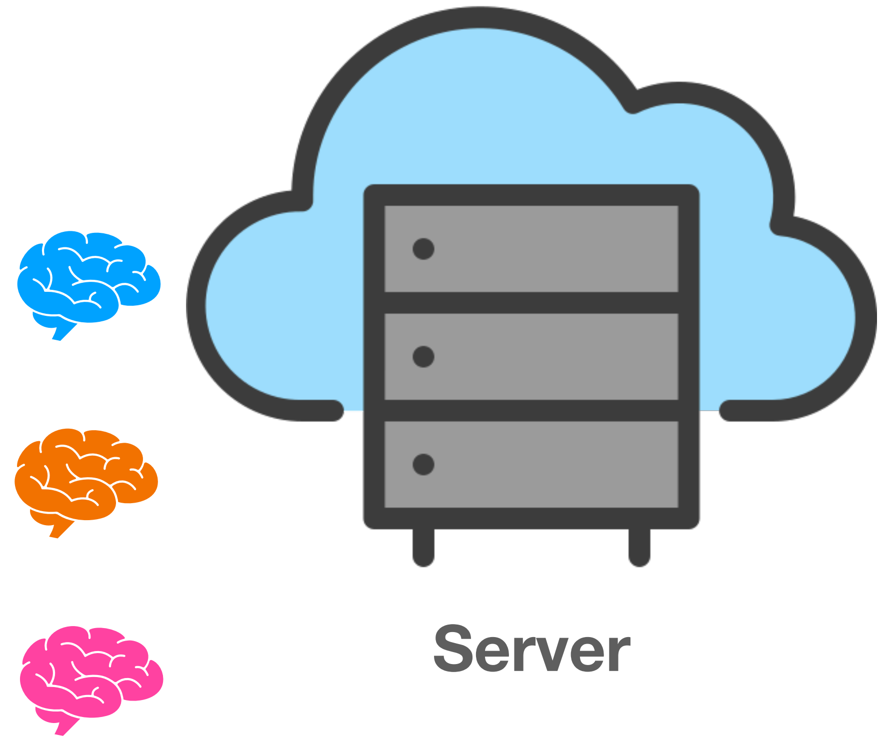
Dynamic communication leaks privacy



Communication schedule for silos in SOTA 📩

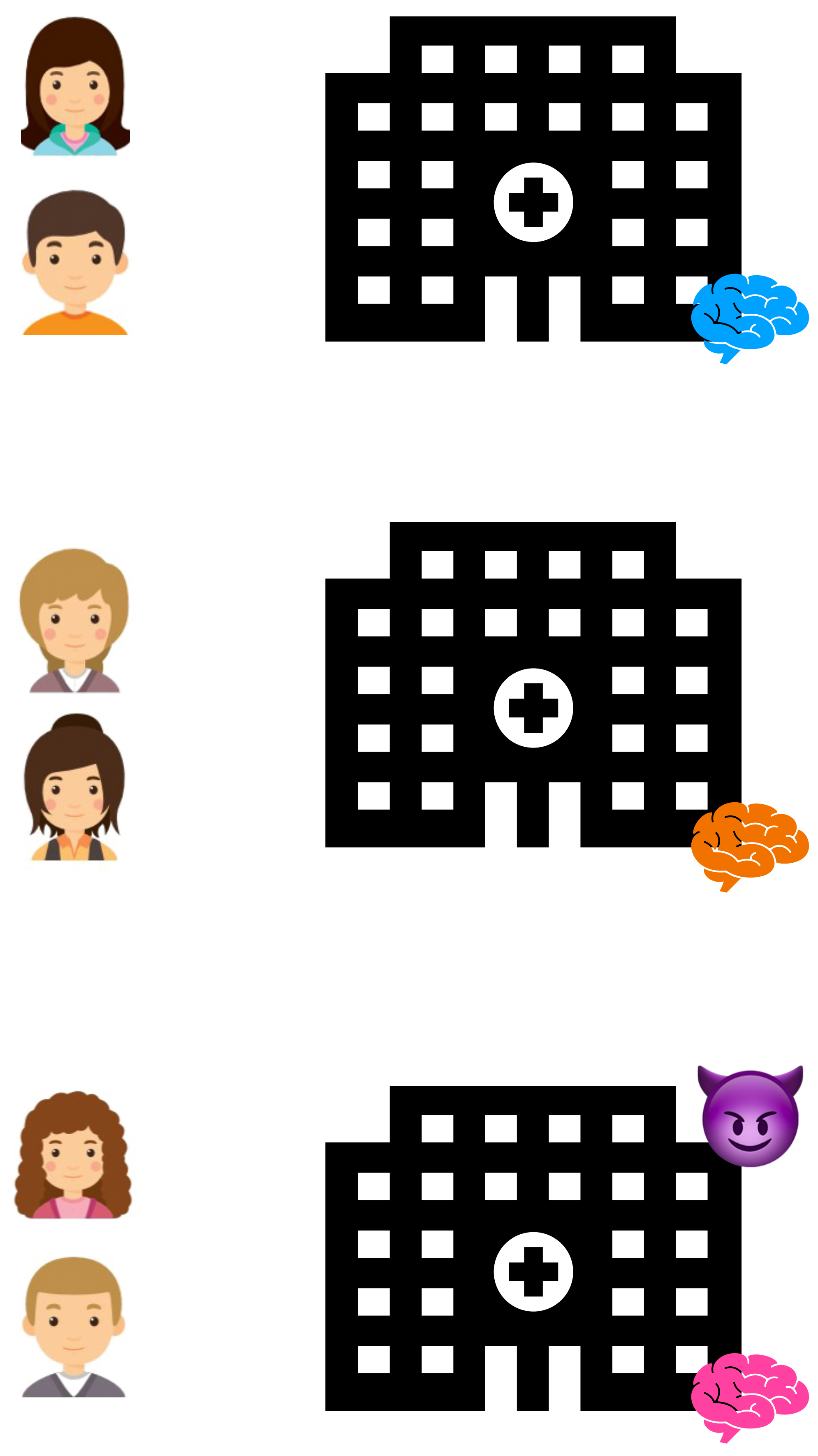
$$\exists i \in [M], \quad f(X_i, Z) > 0$$

- Z — all previous sync data among all silos
- X_i — **new non-private** local data at silo i since recent sync
- f — sync function, shared among all silos



Privacy Gap in SOTA

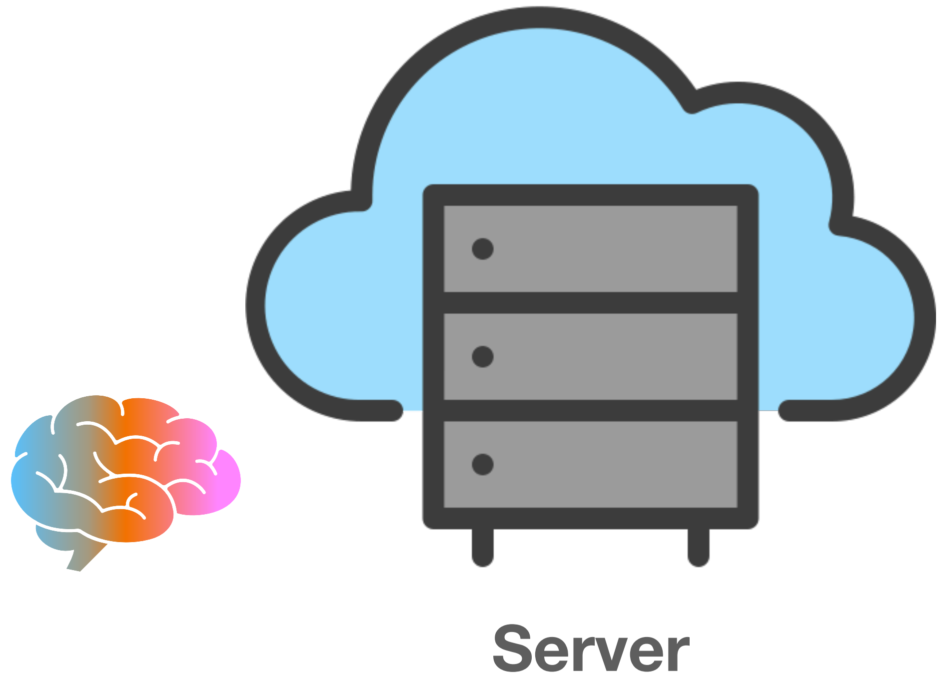
Dynamic communication leaks privacy



Communication schedule for silos in SOTA 📩

$$\exists i \in [M], \quad f(X_i, Z) > 0$$

- Z — all previous sync data among all silos
- X_i — **new non-private** local data at silo i since recent sync
- f — sync function, shared among all silos



Malicious silo can take advantage of this to infer user's sensitive data in another silo

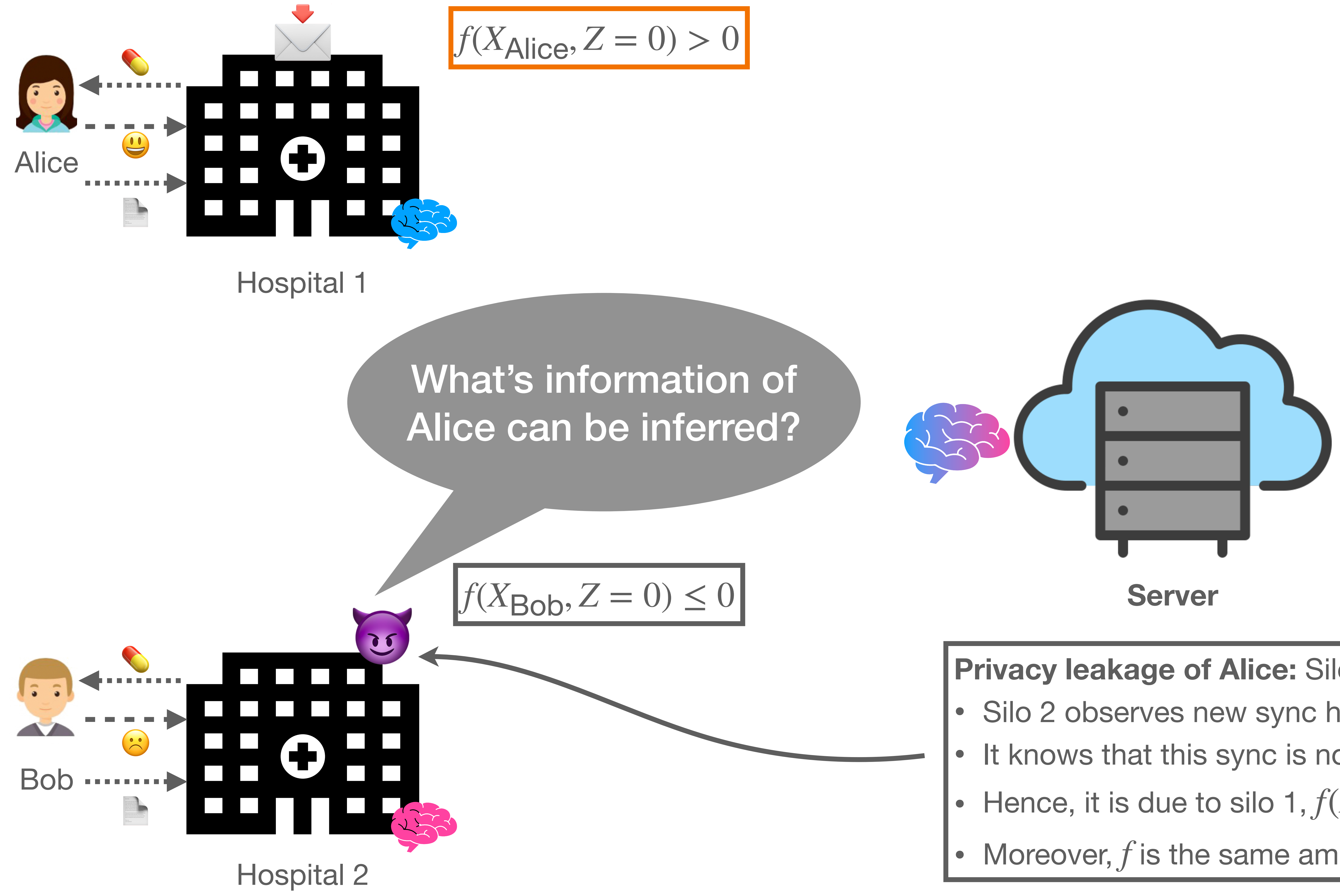
Privacy Gap in SOTA

A simple toy-example attack

Communication schedule for silos in SOTA

$$\exists i \in [M], \quad f(X_i, Z) > 0$$

- Z — all previous sync data among all silos
- X_i — **new non-private** local data at silo i since recent sync
- f — sync function, shared among all silos



time $t = 1$

Privacy leakage of Alice: Silo 2 can infer Alice's data

- Silo 2 observes new sync happens
- It knows that this sync is not triggered by itself
- Hence, it is due to silo 1, $f(X_{\text{Alice}}, Z=0) > 0$
- Moreover, f is the same among silos

Privacy Gap in SOTA

A simple toy-example attack

Communication schedule for silos in SOTA

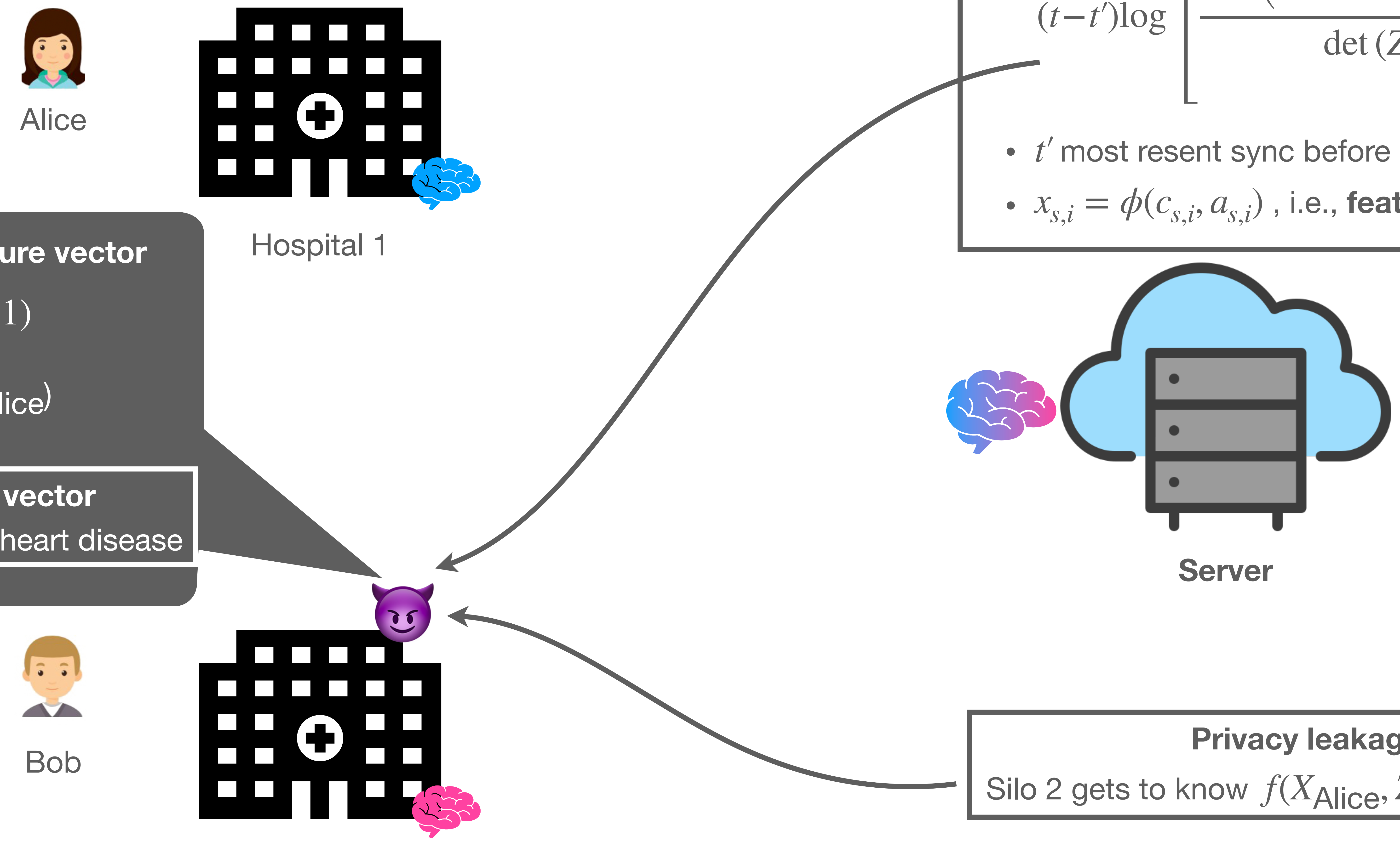
$$\exists i \in [M], \quad f(X_i, Z) > 0$$

- Z — all previous sync data among all silos
- X_i — **new non-private** local data at silo i since recent sync
- f — sync function, shared among all silos

In particular, a sync triggered by silo i at time t if

$$(t-t')\log\left[\frac{\det\left(Z+\sum_{s=t'+1}^{t} x_{s,i}x_{s,i}^{\top}+\lambda I\right)}{\det(Z+\lambda I)}\right] > D$$

- t' most resent sync before t and D some threshold
- $x_{s,i} = \phi(c_{s,i}, a_{s,i})$, i.e., **feature vector**



Silo 2 knows the norm of Alice's feature vector

$$\| x_{\text{Alice}} \|^2 > C := \lambda(e^D - 1)$$

$$(t = 1, t' = 0, Z = 0, x_{1,1} = x_{\text{Alice}})$$

Context info leaked via feature vector

i.e., Alice may have both diabetes and heart disease

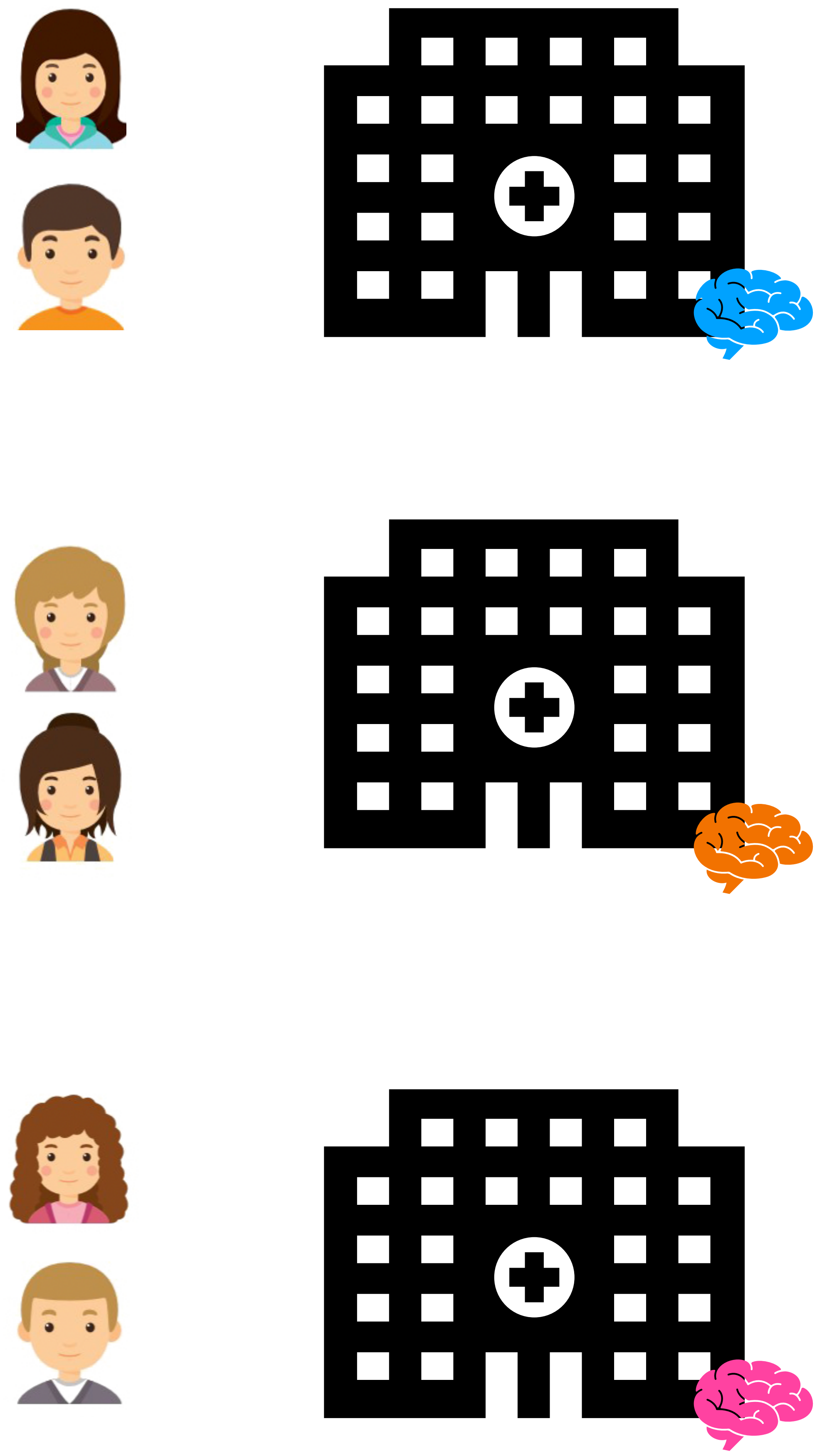
Privacy leakage of Alice

Silo 2 gets to know $f(X_{\text{Alice}}, Z = 0) > 0$ at time $t = 1$

Regret and Communication Gaps in SOTA

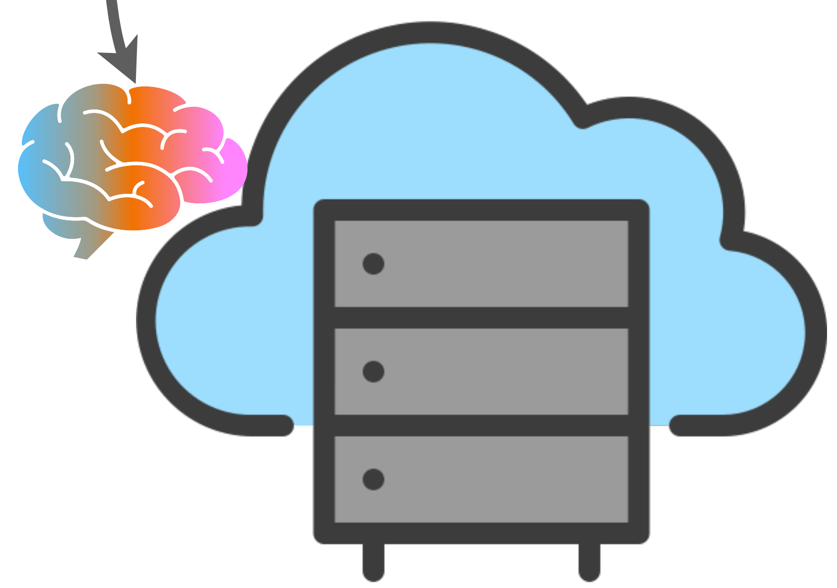
Low communication cost is also ungrounded

i.e., $O(\log T)$??



Larger total privacy noise implies larger regret

Ignore the privacy issue, the total amount of privacy noise in SOTA needs to be $\sigma^2_{\text{total}} = M\,\sigma^2$, i.e., M factor of its current one (recall M is the no. silos)



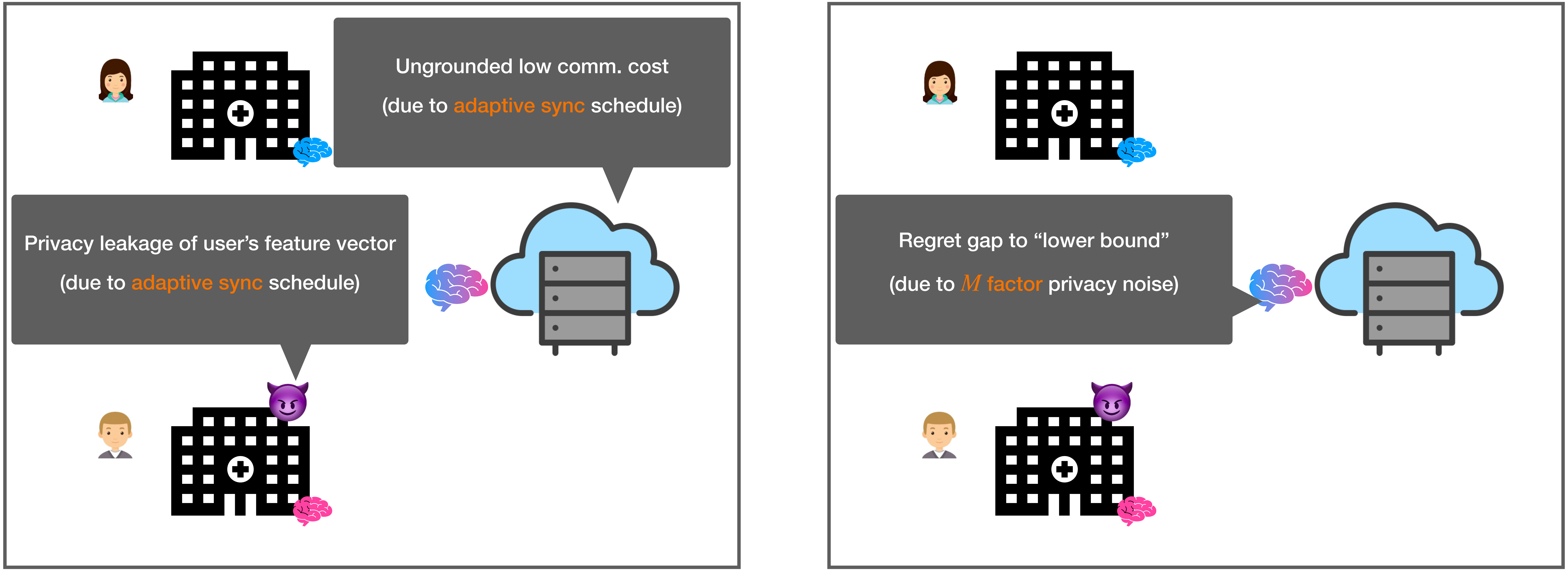
Current conclusion in SOTA becomes ungrounded

After the correction of M factor, the regret due to privacy changes from

$\tilde{O}(\sqrt{MT/\epsilon})$ *(match the "lower bound" of a single agent)*

to

$\tilde{O}(M^{3/4}\sqrt{T/\epsilon})$ *(has a gap of $M^{1/4}$ compared to "lower bound")*



Motivating Questions

1. How to address the privacy leakage? (💡 *a fixed communication schedule may work, i.e., does not depend on user's non-private data)*
2. How to correct the regret bound while preserving the privacy?
3. How to close the gap compared to the "lower bound" ? (💡 *need a way to get rid of M factor)*
4. **If possible, can we achieve all of them via a generic method? (**💡 *a template algorithm with a template proof is preferred***)**

Our Approach

A Generic Algorithm

Private-FedLinUCB

Private-FedLinUCB
(fixed batch sync of LinUCB with privacy)

Parameters: batch size B, privacy protocol $P = (R, S, A)$

Initialize: $\forall i, W_i = 0, U_i = 0$; $\widetilde{W}_{\text{syn}} = 0$, $\widetilde{U}_{\text{syn}} = 0$

for $t = 1, \ldots, T$ **do**

  for each agent $i = 1, \ldots, M$ **do**

    $V_{t,i} = \lambda I + \widetilde{W}_{\text{syn}} + W_i$, $U_{t,i} = \widetilde{U}_{\text{syn}} + U_i$

    Estimate: $\hat{\theta}_{t,i} = V_{t,i}^{-1} U_{t,i}$

    UCB: $a_{t,i} = \arg\max_a \phi(c_{t,i}, a)^\top \hat{\theta}_t + \beta_t \left\| \phi(c_{t,i}, a) \right\|_{V_{t,i}^{-1}}$

    Observe reward $y_{t,i}$; set $x_{t,i} = \phi(c_{t,i}, a_{t,i})$

    Update local data: $W_i = W_i + x_{t,i} x_{t,i}^\top$, $U_i = U_i + x_{t,i} y_{t,i}$

  if $t \bmod B = 0$ **then**

    $\widetilde{W}_{\text{syn}} = P(\{W_i\}_{i \in [M]})$, $\widetilde{U}_{\text{syn}} = P(\{U_i\}_{i \in [M]})$

    Receive $\widetilde{W}_{\text{syn}}$, $\widetilde{U}_{\text{syn}}$ from server

    Reset $W_i = 0, U_i = 0$

W_i — sum of local covariance matrices at agent i
U_i — sum of local bias vectors at agent i
$\widetilde{W}_{\text{syn}}$ — private sync covariance matrices among all agents
$\widetilde{U}_{\text{syn}}$ — private sync bias vectors among all agents

$V_{t,i}$ — sum of regularizer, sync and new local cov. matrices
$U_{t,i}$ — sum of sync and new local bias vectors

$P = (R, S, A)$, a template protocol for **summation** (will discuss it soon)
  R — local randomzier at agent side (on W_i, U_i)
  S — shuffler or identity mapping, between agents, server
  A — analyzer at server side

Single agent LinUCB[APS11] 101

For $t = 1, \ldots, T$:

1. **Estimate** θ^*: $\hat{\theta}_t = V_t^{-1} U_t$,

   $(V_t = \lambda I + \sum_{s=1}^{t-1} x_s x_s^\top$ ("covariance"), $U_t = \sum_{s=1}^{t-1} x_s y_s$ ("bias"))

2. **UCB:** $a_t = \arg\max_a \phi(c_t, a)^\top \hat{\theta}_t + \beta_t \left\| \phi(c_t, a) \right\|_{V_t^{-1}}$

   $(x_t = \phi(c_t, a_t)$, β_t — chosen via confidence bound)

A Generic Algorithm

Private-FedLinUCB

Private-FedLinUCB
(fixed batch sync of LinUCB with privacy)

Parameters: batch size B, privacy protocol $P = (R, S, A)$

Initialize: $\forall i, W_i = 0, U_i = 0$; $\widetilde{W}_{\text{syn}} = 0$, $\widetilde{U}_{\text{syn}} = 0$

for $t = 1,\ldots,T$ **do**

  for each agent $i = 1,\ldots,M$ **do**

    $V_{t,i} = \lambda I + \widetilde{W}_{\text{syn}} + W_i$, $U_{t,i} = \widetilde{U}_{\text{syn}} + U_i$

    Estimate: $\hat{\theta}_{t,i} = V_{t,i}^{-1} U_{t,i}$

    UCB: $a_{t,i} = \arg\max_a \phi(c_{t,i}, a)^\top \hat{\theta}_t + \beta_t \left\| \phi(c_{t,i}, a) \right\|_{V_{t,i}^{-1}}$

    Observe reward $y_{t,i}$; set $x_{t,i} = \phi(c_{t,i}, a_{t,i})$

    Update local data: $W_i = W_i + x_{t,i} x_{t,i}^\top$, $U_i = U_i + x_{t,i} y_{t,i}$

  if t mod $B = 0$ **then**

    $\widetilde{W}_{\text{syn}} = P(\{W_i\}_{i \in [M]})$, $\widetilde{U}_{\text{syn}} = P(\{U_i\}_{i \in [M]})$

    Receive $\widetilde{W}_{\text{syn}}$, $\widetilde{U}_{\text{syn}}$ from server

    Reset $W_i = 0, U_i = 0$

Single agent LinUCB[APS11] 101

For $t = 1,\ldots,T$:

1. **Estimate θ^*:** $\hat{\theta}_t = V_t^{-1} U_t$,

   $(V_t = \lambda I + \sum_{s=1}^{t-1} x_s x_s^\top$ ("covariance"), $U_t = \sum_{s=1}^{t-1} x_s y_s$ ("bias"))

2. **UCB:** $a_t = \arg\max_a \phi(c_t, a)^\top \hat{\theta}_t + \beta_t \left\| \phi(c_t, a) \right\|_{V_t^{-1}}$

   $(x_t = \phi(c_t, a_t)$, β_t — chosen via confidence bound)

Remark: fixed vs. adaptive schedule

- It now suffices to privatize each sent messages for silo-level LDP guarantee
  - $R(W_i)$ and $R(U_i)$ is private at each sync round $k \in [T/B]$
  - without worrying privacy leakage via schedule
  - B needs to balance between comm. cost, regret, and privacy
- The problem in SOTA is: schedule depends on **non-private data** (i.e., W_i)
  - how about privatizing it first and then be adaptive?
  - we show that it will lead to fundamental challenge in regret analysis

$P = (R, S, A)$,a template protocol for **summation** (will discuss it soon)
R — local randomzier at agent side (on W_i, U_i)
S — shuffler or identity mapping, between agents, server
A — analyzer at server side

A Generic Priv. Protocol

Distributed Tree-based alg.

P = (R,S,A), privacy protocol
(distributed version of Tree-based algorithm)

Differential Privacy 201

1. **Gaussian mechanism** for private sum of l_2 bounded vectors

   i.e., $\widetilde{s}$ is the private sum of $\sum_{s=1}^{k} \gamma_s$ under (ϵ, δ)-DP

   $$\widetilde{s} = \sum_{s=1}^{k} \gamma_s + \mathcal{N}(0, \sigma^2 I),\ \sigma^2 \approx \frac{L^2 \log(1/\delta)}{\epsilon^2}$$

   Intuition: change one data, the sum changes in l_2, bounded by L

2. **Continual private sum** (essential for private online learning)

   i.e., a stream of data $\gamma_1, \ldots, \gamma_K$, compute $\widetilde{s}_k$ — priv. sum of $\sum_{s=1}^{k} \gamma_s$

   Simple Approach I: add noise ($\approx 1/\epsilon^2$) to each γ_s

   — (ϵ, δ)-DP (by post-processing)

   — total noise is K/ϵ^2 (**!**)

   Simple Approach II: add noise ($\approx 1/\epsilon^2$) to each prefix sum

   — total noise is $1/\epsilon^2$ for all k

   — $\approx (\sqrt{K}\epsilon, \delta')$-DP (by composition of DP)

   — i.e., for (ϵ, δ)-DP, the total noise needs to be K/ϵ^2 (**!**)

A Generic Priv. Protocol

Distributed Tree-based alg.

P = (R,S,A), privacy protocol
(distributed version of Tree-based algorithm)

Differential Privacy 201

1. **Gaussian mechanism** for private sum of l_2 bounded vectors

   i.e., $\widetilde{s}$ is the private sum of $\sum_{i=1}^{n} \gamma_i$ under (ϵ, δ)-DP

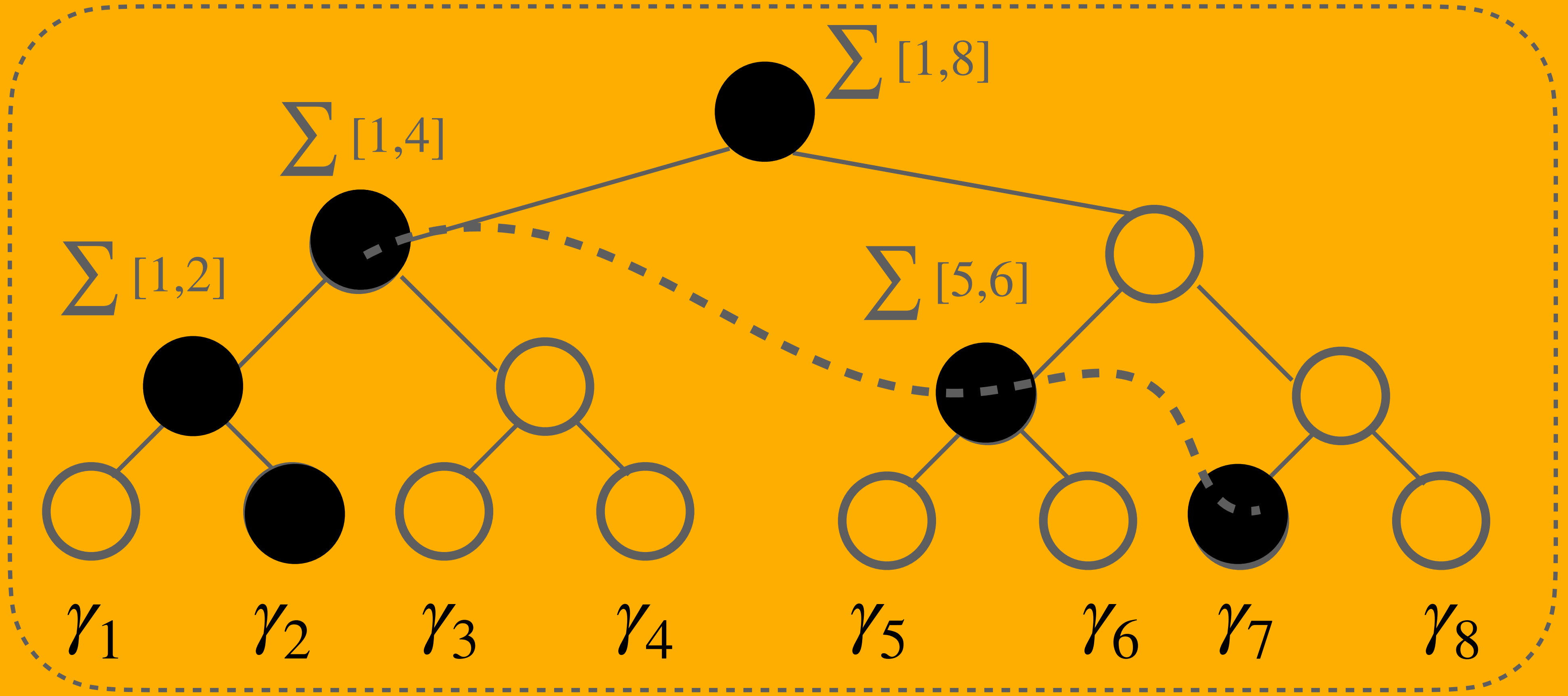
   $$\widetilde{s} = \sum_{i=1}^{n} \gamma_i + \mathcal{N}(0, \sigma^2 I),\ \sigma^2 \approx \frac{L \log(1/\delta)}{\epsilon^2}$$

   Intuition: change one data, the sum changes in l_2, bounded by L

2. **Continual private sum** (essential for private online learning)

   i.e., a stream of data $\gamma_1, \ldots, \gamma_K$, compute $\widetilde{s}_t$ — priv. sum of $\sum_{s=1}^{k} \gamma_s$

   Tree-based algorithm [CSS11]: add noise to partial sum $\sum [i, j]$



Key observations:

— each data affects at most $O(\log K)$ p-sums ($\widetilde{O}(1/\epsilon^2)$ noise each)

— each prefix sum needs at most $O(\log K)$ partial-sums (p-sums)

— total noise is still $\widetilde{O}(1/\epsilon^2)$ (✅ ignore log factor)

A Generic Priv. Protocol

Distributed Tree-based alg.

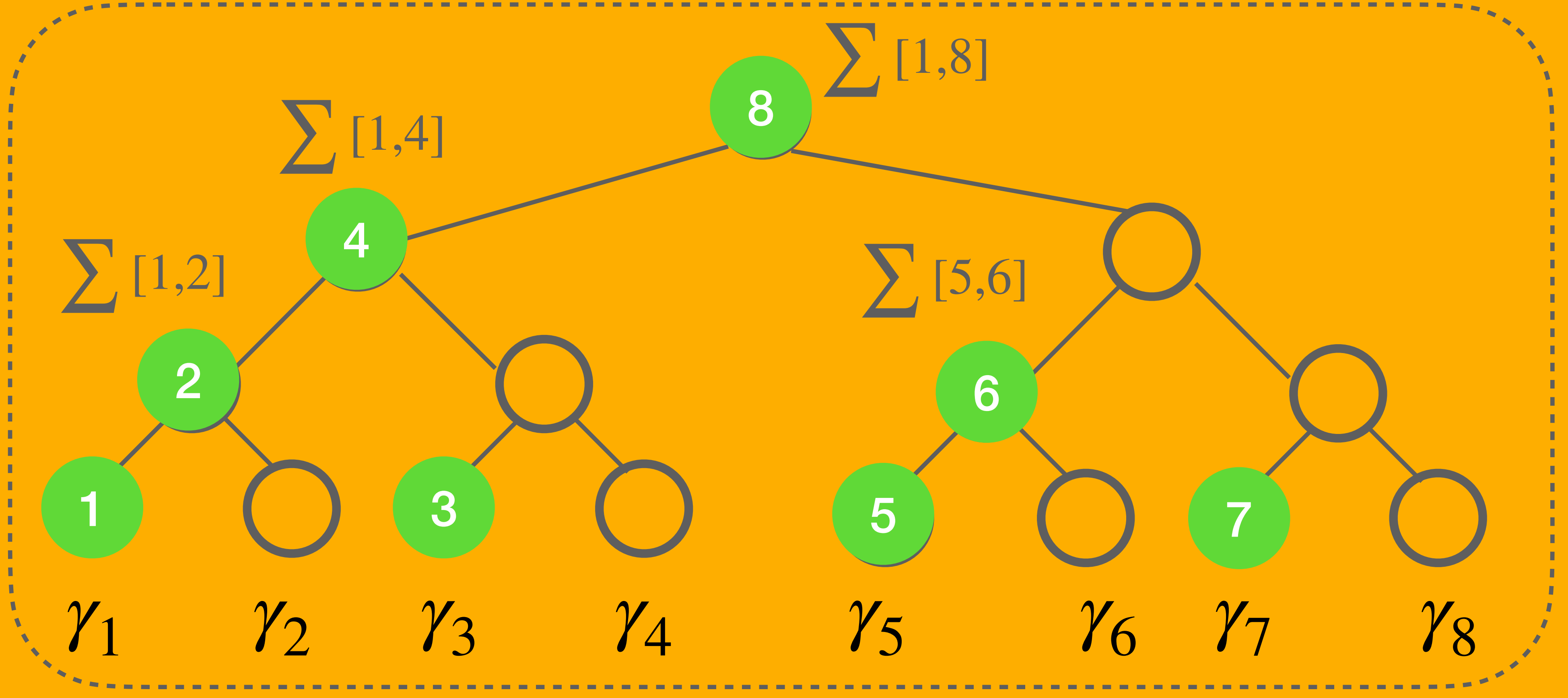
`P = (R,S,A), privacy protocol`

(distributed version of Tree-based algorithm)

Differential Privacy 201

Tree-based algorithm [CSS11] for continual private sum

i.e., a stream of data $\gamma_1, \ldots, \gamma_K$, compute $\tilde{s}_t$ — priv. sum of $\sum_{s=1}^{k} \gamma_s$



Sequential implementation (dynamically compute p-sum, tree node)

1. For each round k, express k in binary form: $k = \sum_j \text{Bin}_j(k) \cdot 2^j$
   (e.g., for $k = 6$, it is 110)
2. Find index of first one $i_k = \min\{j : \text{Bin}_j(k) = 1\}$ (for $k = 6$, $i_k = 1$)
3. Compute non-private p-sum: $\alpha_{i_k} = \sum_{j<i_k} \alpha_j + \gamma_k$
   (α_j — stores the sum of 2^j data)
4. Private p-sum $\tilde{\alpha}_{i_k} = \alpha_{i_k} + \mathcal{N}(0, \sigma^2 I)$
5. Final output $\tilde{s}_k = \sum_{j:\text{Bin(k)}=1} \tilde{\alpha}_j$ (for $k = 6$, [1,4] + [5,6])

A Generic Priv. Protocol

Distributed Tree-based alg.

P = (R,S,A), privacy protocol
(distributed version of Tree-based algorithm)

Procedure: Local Randomizer R at each agent $i \in [M]$

for each sync $k = 1,\ldots,K$ **do**

Express k in binary form: $k = \sum_j \text{Bin}_j(k) \cdot 2^j$

Find index of first one $i_k = \min\{j : \text{Bin}_j(k) = 1\}$

Compute non-private p-sum: $\alpha_{i_k} = \sum_{j<i_k} \alpha_j + \gamma_k$

Output noisy p-sum $\widetilde{\alpha}_{i_k,i} = \alpha_{i_k} + \mathcal{N}(0,\sigma^2 I)$

Procedure: Shuffler S (could be empty or identity mapping)

Procedure: Analyzer A at server

for each sync $k = 1,\ldots,K$ **do**

Express k in binary form and find index of first one i_k

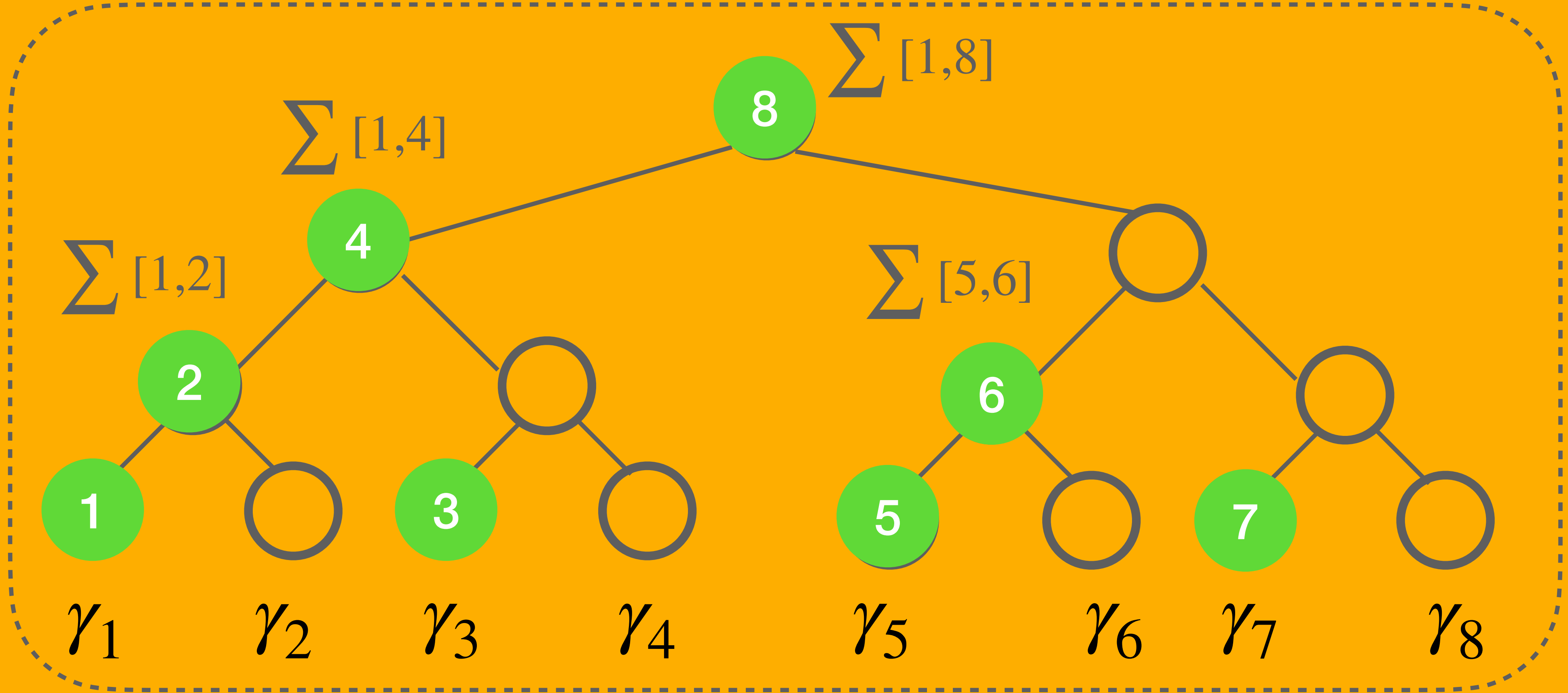
Add noisy p-sums from **all agents** $\widetilde{\alpha}_{i_k} = \sum_{i\in[M]} \widetilde{\alpha}_{i_k,i}$

Output total sum: $\widetilde{s}_k = \sum_{j:\text{Bin(k)}=1} \widetilde{\alpha}_j$

Differential Privacy 201

Tree-based algorithm [CSS11] for continual private sum

i.e., a stream of data $\gamma_1, \ldots, \gamma_K$, compute $\widetilde{s}_t$ — priv. sum of $\sum_{s=1}^{k} \gamma_s$



Sequential implementation (dynamically compute p-sum, tree node)

1. For each round k, express k in binary form: $k = \sum_j \text{Bin}_j(k) \cdot 2^j$
   (e.g., for $k = 6$, it is 110)
2. Find index of first one $i_k = \min\{j : \text{Bin}_j(k) = 1\}$ (for $k = 6$, $i_k = 1$)
3. Compute non-private p-sum: $\alpha_{i_k} = \sum_{j<i_k} \alpha_j + \gamma_k$
   (α_j — stores the sum of 2^j data)
4. Private p-sum $\widetilde{\alpha}_{i_k} = \alpha_{i_k} + \mathcal{N}(0,\sigma^2 I)$
5. Final output $\widetilde{s}_k = \sum_{j:\text{Bin(k)}=1} \widetilde{\alpha}_j$ (for $k = 6$, [1,4] + [5,6])

Private-FedLinUCB

(fixed batch sync of LinUCB with privacy)

Parameters: batch size B, privacy protocol $P = (R, S, A)$

Initialize: $\forall i, W_i = 0, U_i = 0$; $\widetilde{W}_{\text{syn}} = 0$, $\widetilde{U}_{\text{syn}} = 0$

for $t = 1, \ldots, T$ **do**

for each agent $i = 1, \ldots, M$ **do**

$V_{t,i} = \lambda I + \widetilde{W}_{\text{syn}} + W_i$, $U_{t,i} = \widetilde{U}_{\text{syn}} + U_i$

Estimate: $\hat{\theta}_{t,i} = V_{t,i}^{-1} U_{t,i}$

UCB: $a_{t,i} = \arg\max_a \phi(c_{t,i}, a)^\top \hat{\theta}_t + \beta_t \left\| \phi(c_{t,i}, a) \right\|_{V_{t,i}^{-1}}$

Observe reward $y_{t,i}$; set $x_{t,i} = \phi(c_{t,i}, a_{t,i})$

Update local data: $W_i = W_i + x_{t,i} x_{t,i}^\top$, $U_i = U_i + x_{t,i} y_{t,i}$

if $t \bmod B = 0$ **then**

$\widetilde{W}_{\text{syn}} = P(\{W_i\}_{i \in [M]})$, $\widetilde{U}_{\text{syn}} = P(\{U_i\}_{i \in [M]})$

Receive $\widetilde{W}_{\text{syn}}$, $\widetilde{U}_{\text{syn}}$ from server

Reset $W_i = 0, U_i = 0$

$$\gamma_k^{\text{cov}} = \sum_{t=(k-1)B+1}^{kB} x_t x_t^\top$$

$$\gamma_k^{\text{bias}} = \sum_{t=(k-1)B+1}^{kB} x_t y_t$$

P = (R,S,A), privacy protocol

(distributed version of Tree-based algorithm)

Procedure: Local Randomizer R at each agent $i \in [M]$

for each sync $k = 1, \ldots, K$ **do**

Express k in binary form: $k = \sum_j \text{Bin}_j(k) \cdot 2^j$

Find index of first one $i_k = \min\{j : \text{Bin}_j(k) = 1\}$

Compute non-private p-sum: $\alpha_{i_k} = \sum_{j < i_k} \alpha_j + \gamma_k$

Output noisy p-sum $\widetilde{\alpha}_{i_k, i} = \alpha_{i_k} + \mathcal{N}(0, \sigma^2 I)$

Procedure: Analyzer A at server

for each sync $k = 1, \ldots, K$ **do**

Express k in binary form and find index of first one i_k

Add noisy p-sums from **all agents** $\widetilde{\alpha}_{i_k} = \sum_{i \in [M]} \widetilde{\alpha}_{i_k, i}$

Output total sum: $\widetilde{s}_k = \sum_{j : \text{Bin(k)} = 1} \widetilde{\alpha}_j$

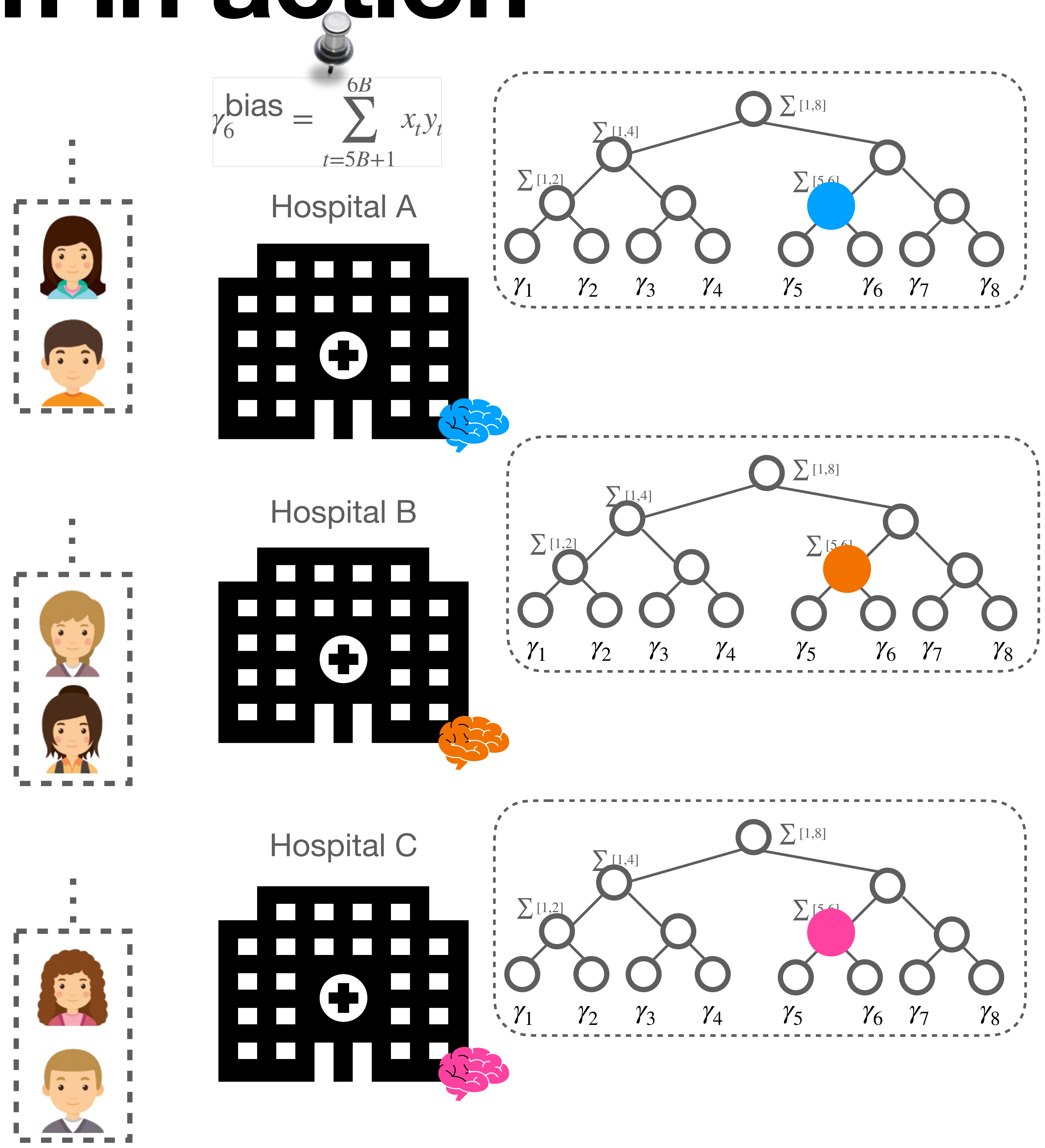
Putting them together

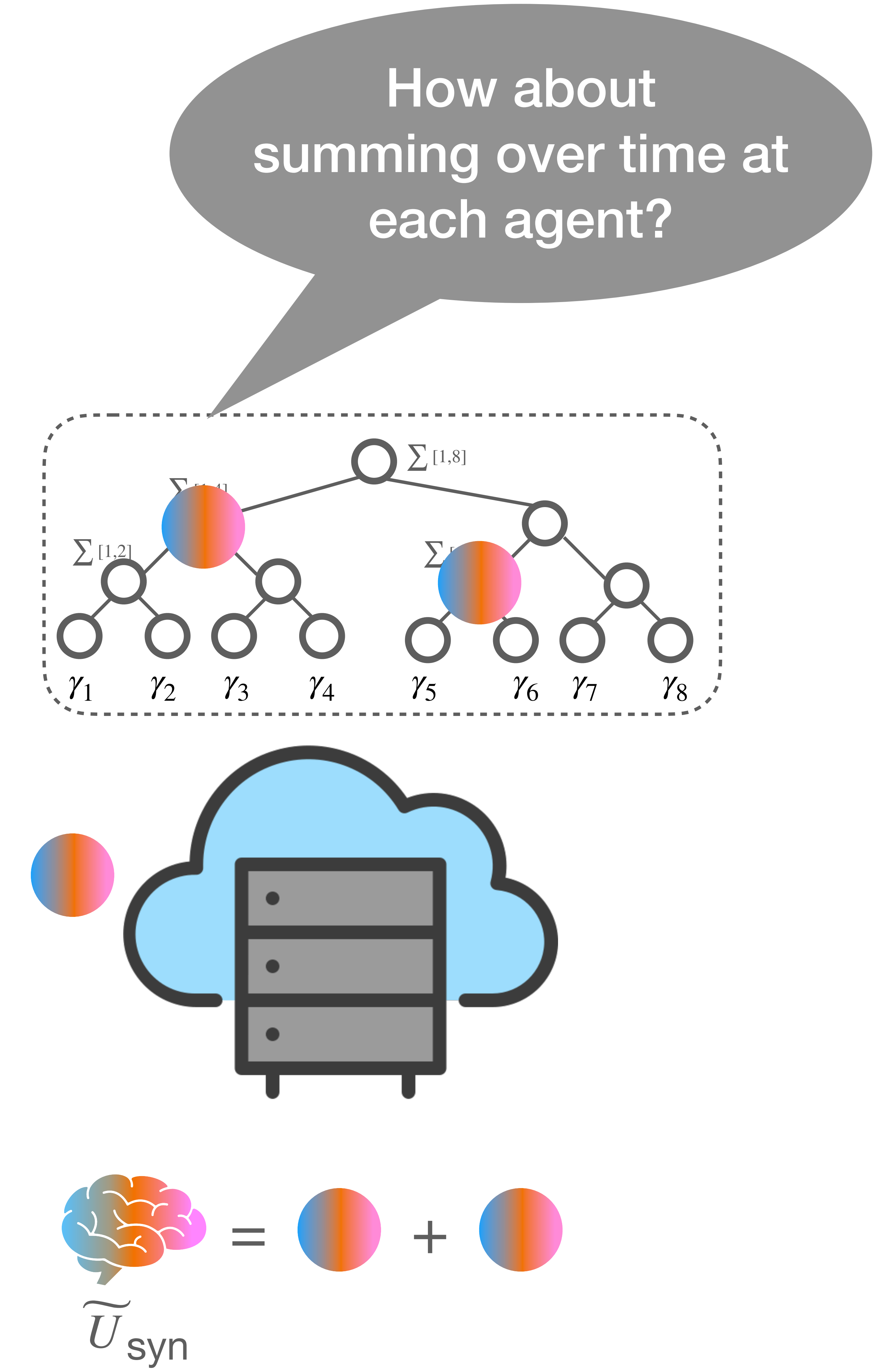
- Each agent runs two privacy protocol — sum of **covariance** matrices (i.e., W_i) and sum of **bias** vectors (i.e., U_i)
- The datapoint γ_k is a **batch** of data — total matrices or vectors during the kth batch
- **The sensitivity does not scale with respect to B**

Algorithm in action

Illustration

time $t = 6B$



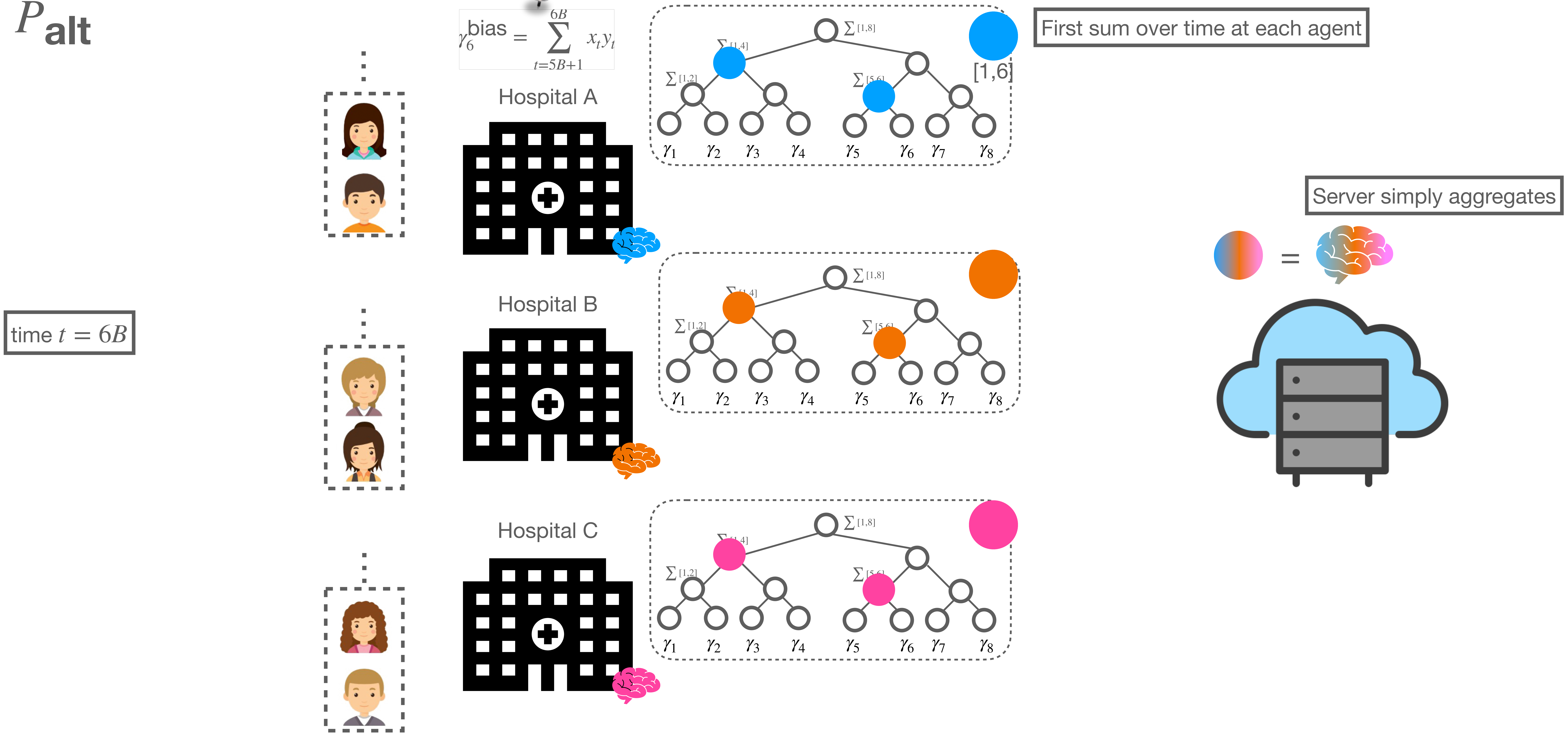


Private sum across both time and agents

$$\sum_{i=1}^{M} \sum_{t=1}^{6B} x_{t,i} y_{t,i}$$

An alternative protocol

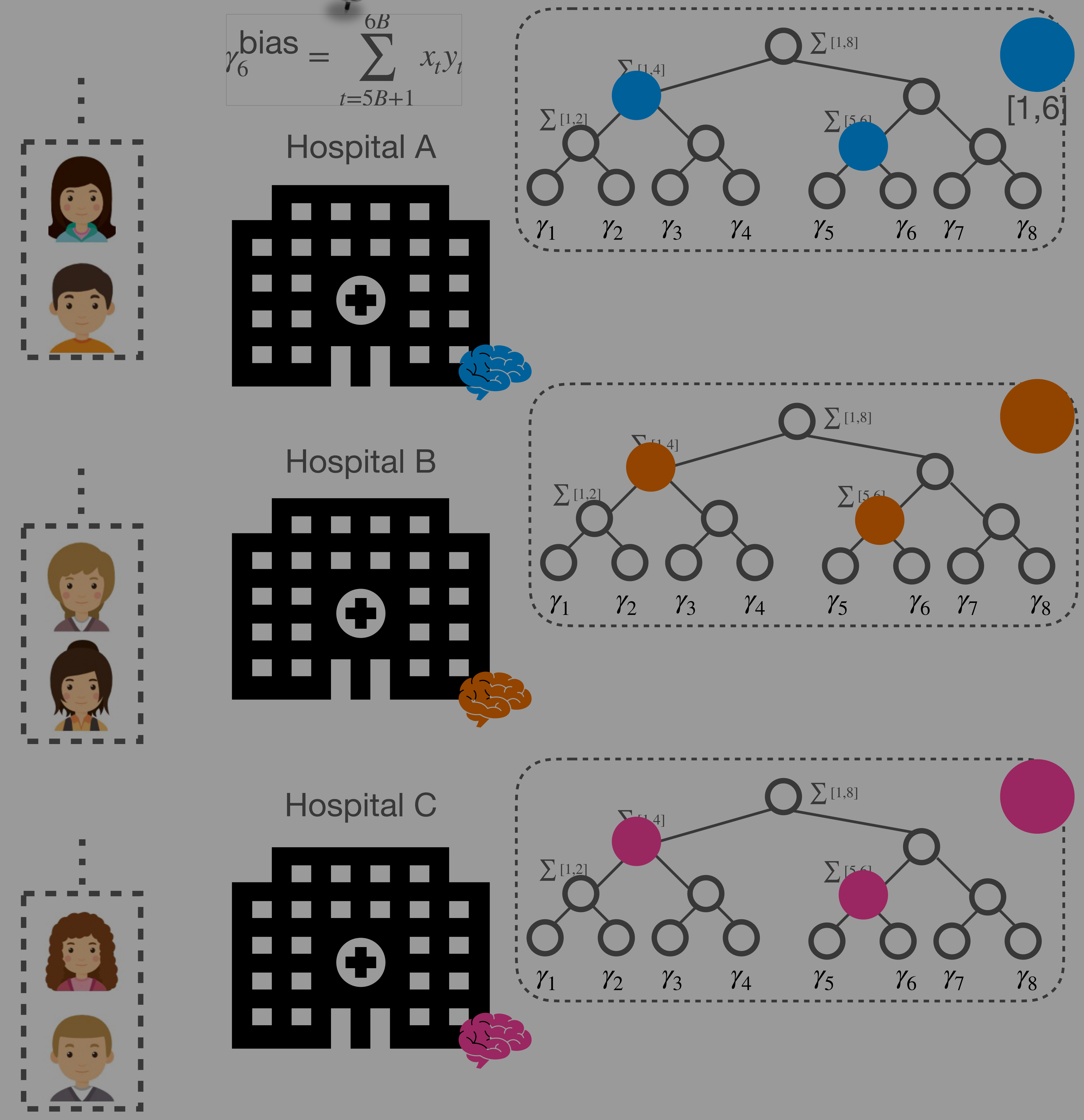
P_{alt}



An alternative protocol

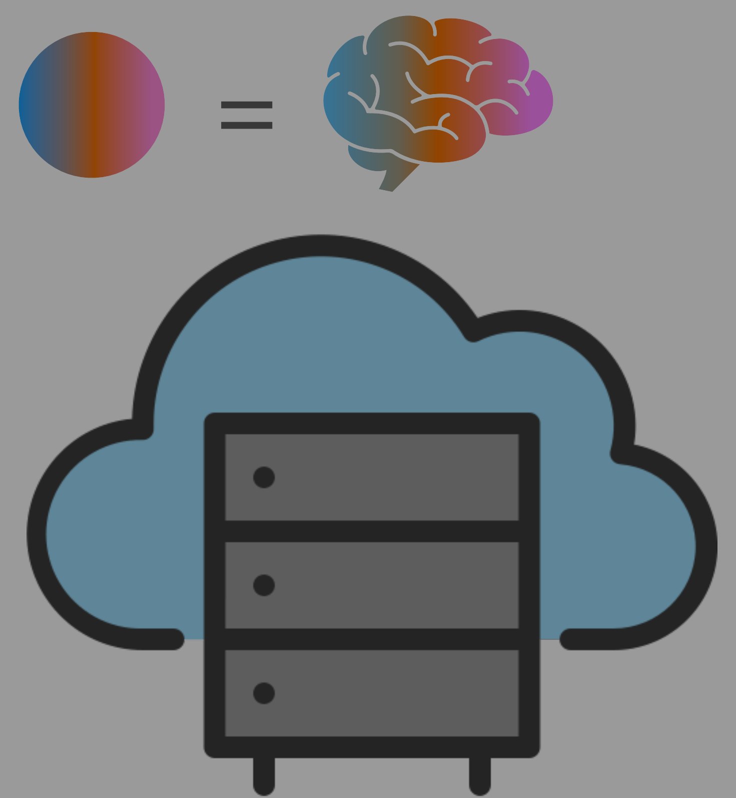
P_{alt}

time $t = 6B$



First sum over time at each agent

Server simply aggregates



Remark: comparisons

- As we will see, both protocols work for silo-level LDP
  - same regret under same (ϵ, δ)-DP
- However, for shuffle DP, things are different
  - our protocol manages to close the gap
  - P_{alt} fails to close the gap (more on this later…😉)

Theoretical Results

Federated LCBs under Silo-level LDP

Fix the issues in SOTA

Theorem 1 (Performance under silo-level LDP, informal)

Let batch size $B = \sqrt{T/M}$, privacy noise in P be $\sigma^2 = 8\kappa \cdot \dfrac{\log(2/\delta) + \epsilon}{\epsilon^2}$ with $\kappa = 1 + \log(T/B)$. Then, `Private-FedLinUCB` enjoys

1. **Privacy** — (ϵ, δ)-silo-level LDP for any $\epsilon > 0,\ \delta \in (0,1)$
2. **Regret** — $R_M(T) =$ non-private regret $+ \sqrt{T}\dfrac{(Md)^{3/4}\log^{1/4}(1/\delta)}{\sqrt{\epsilon}}$
3. **Communication** — $\sqrt{MT}$ rounds of sync between agents and server

Remark: comparisons with related work

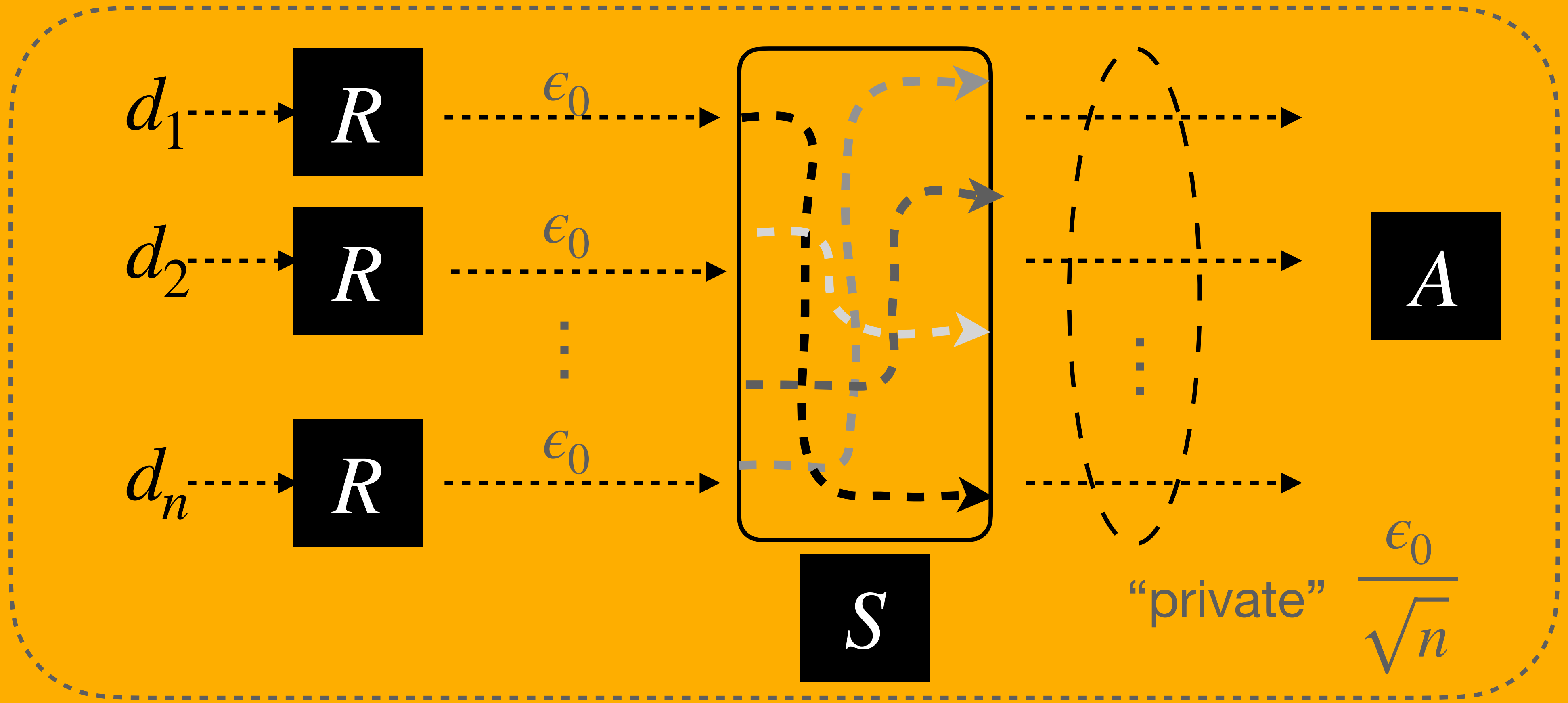
1. **Compared with SOTA[DP20]**
   — privacy: we fix the privacy leakage, thanks to the fixed-batch schedule and tree-based algorithm
   — regret: we establish the correct privacy cost, i.e., the additional regret due to privacy now scales with $M^{3/4}$ (instead of $\sqrt{M}$)
   — communication: communication is worse than SOTA ($\sqrt{T} vs . \log T$) due to fixed-batch comm. But, note that there exists privacy leakage
2. **Compared with "super" single agent under central DP[SS18]**
   — our regret is $M^{1/4}$ factor worse than this "lower bound"

Federated LCBs under SDP

Match the “lower bound”

Differential Privacy 501

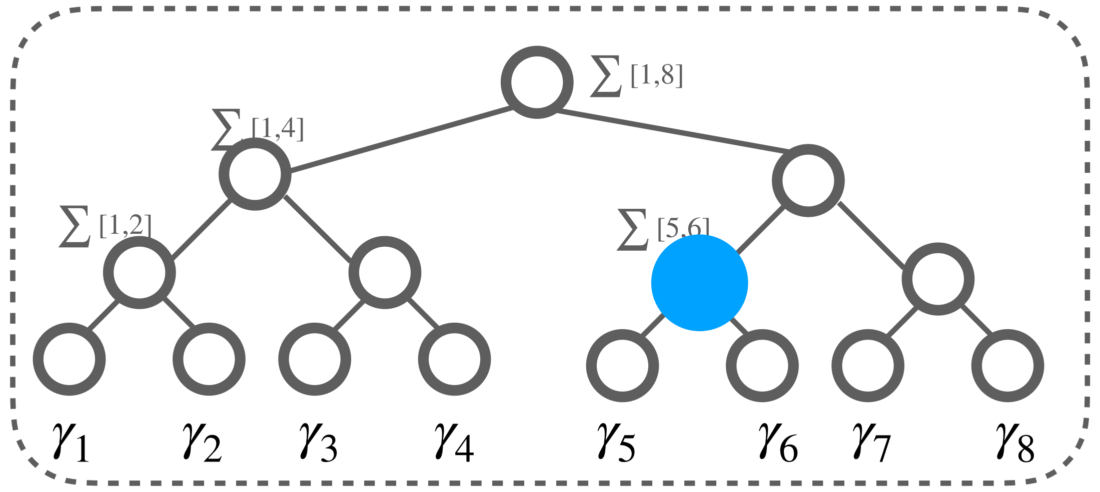
1. **What is shuffle DP (SDP)?**
   — formally defined in [CSUZZ19]
   — $P = (R, S, A)$, “the output of shuffler is private”
   — (change any d_i, the outputs are “close”)



2. **How to achieve it?**
   — one way is via LDP amplification, e.g., [FMT20]
   — shuffle n LDP outputs (each ϵ_0-DP), then it is $\approx \epsilon_0/\sqrt{n}$ SDP
   — **“reduce the privacy loss by a factor of $1/\sqrt{n}$ ✅”**
   — (intuition: hiding among clones)

Federated LCBs under SDP

Match the "lower bound"



How about adding shuffler between agents and server? $1/\sqrt{M}$

Good news: this amplification can close the gap ✅

Bad news: one cannot directly use existing results ❗

— they only amplify LDP (R oper. on single data)

— in our case, R oper. on multiple datapoints

— (this leads to key difference in the analysis)

Clones are harder to create due to multiple local points

A new amplification lemma is derived ✅

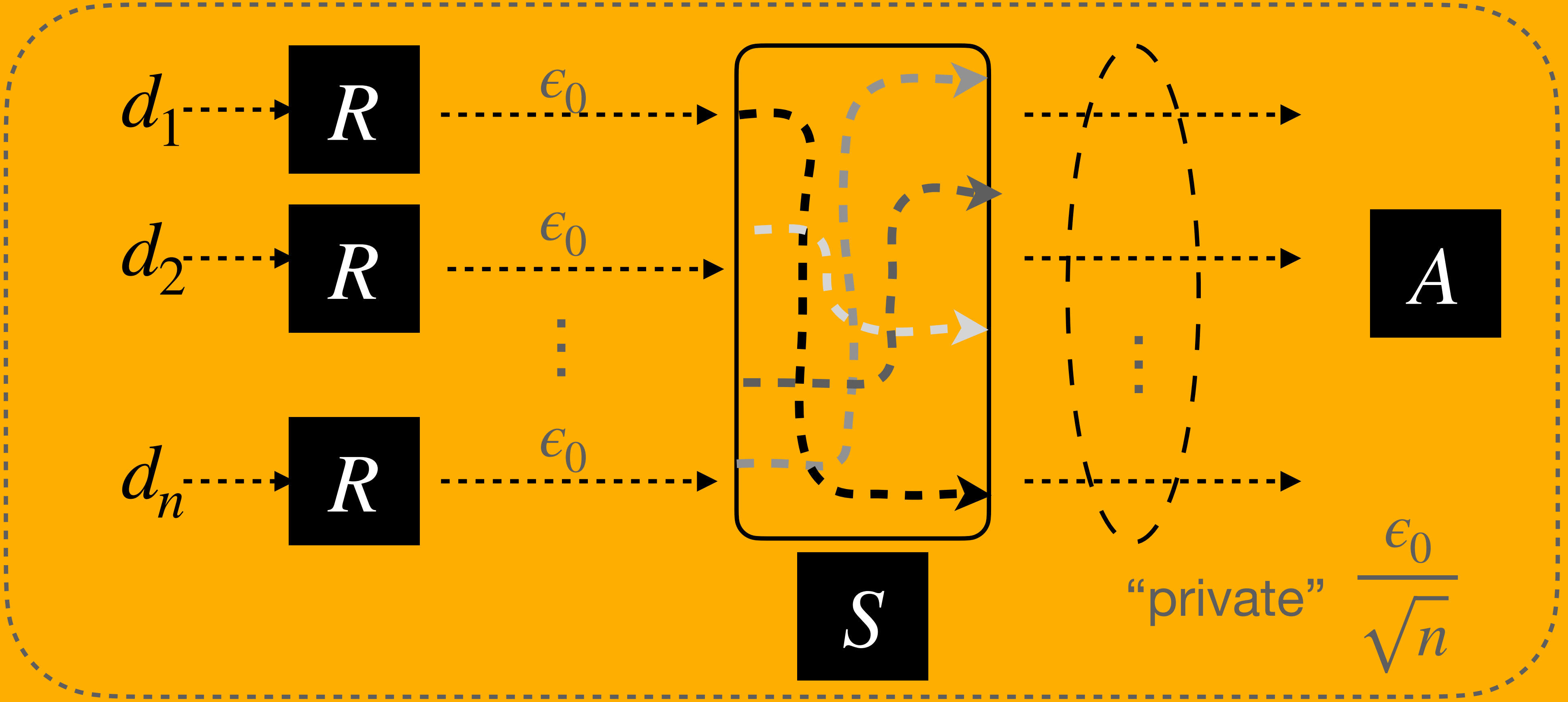
— tailored for Gaussian DP mecha.

— avoid group privacy

— control the blow up in δ

Differential Privacy 501

1. **What is shuffle DP (SDP)?**
   — formally defined in [CSUZZ19]
   — $P = (R, S, A)$, "the output of shuffler is private"
   — (change any d_i, the outputs are "close")



2. **How to achieve it?**
   — one way is via LDP amplification, e.g., [FMT20]
   — shuffle n LDP outputs (each ϵ_0-DP), then it is $\approx \epsilon_0/\sqrt{n}$ SDP
   — **"reduce the privacy loss by a factor of $1/\sqrt{n}$ ✅"**
   — (intuition: hiding among clones)

Federated LCBs under SDP

Match the “lower bound”

How to improve the privacy guarantees?

Theorem 2 (Performance under SDP, informal)

Let batch size $B = \sqrt{T/M}$ and $\kappa = 1 + \log(T/B)$, privacy noise in P be $\sigma^2 = \tilde{O}\left(\frac{\kappa \log(1/\delta)}{\epsilon^2 M}\right)$. Then, `Private-FedLinUCB (with shuffler)` enjoys

1. **Privacy** — (ϵ, δ)-SDP for any $\epsilon \in \left(0, \frac{\sqrt{\kappa}}{C_1 T \sqrt{M}}\right)$, $\delta \in \left(0, \frac{\kappa}{C_2 T}\right)$, where C_1, C_2 are constants
2. **Regret** — $R_M(T) =$ non-private regret $+ \sqrt{MT}\frac{d^{3/4}\log^{3/4}(M\kappa/\delta)}{\sqrt{\epsilon}}$
3. **Communication** — $\sqrt{MT}$ rounds of sync between agents and server

Match the “lower bound”
Privacy cost is on the order of $\sqrt{MT}$

Privacy holds for small ϵ only
This comes from two factors due to amplification lemma
— $1/\sqrt{M}$ is the standard term
— $1/T$ is the new term due to multiple local points

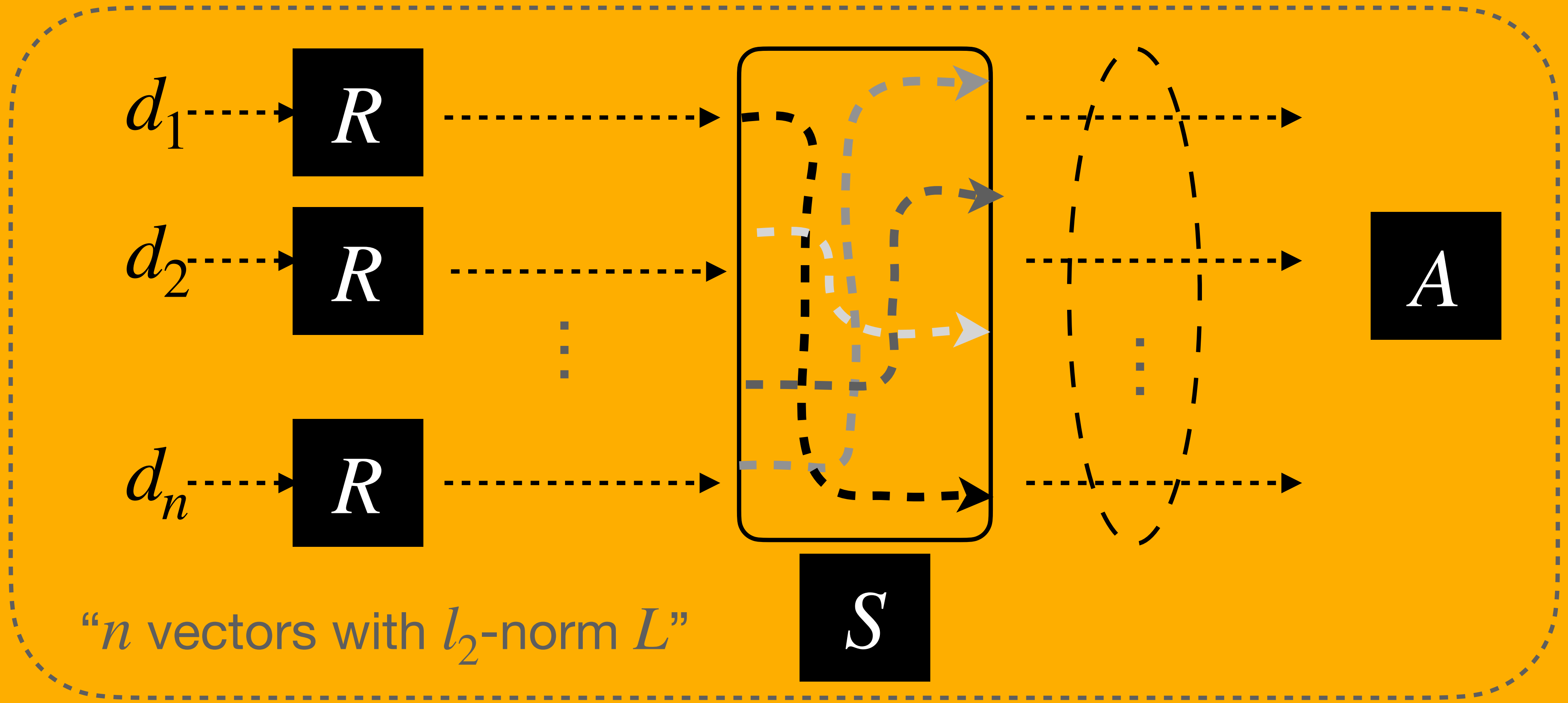
Minimum modifications
Compared to silo-level LDP, one only needs to
— add a shuffler
— adjust the noise in local randomizer R

Federated LCBs under SDP

Leverage vector-sum protocol

Differential Privacy 502

1. **How to achieve SDP?**
   — instead of using amplification lemma
   — one can use specific shuffle protocol
   — $P_{\text{vec}} = (R_{\text{vec}}, S, A_{\text{vec}})$[CJMP21] is one example

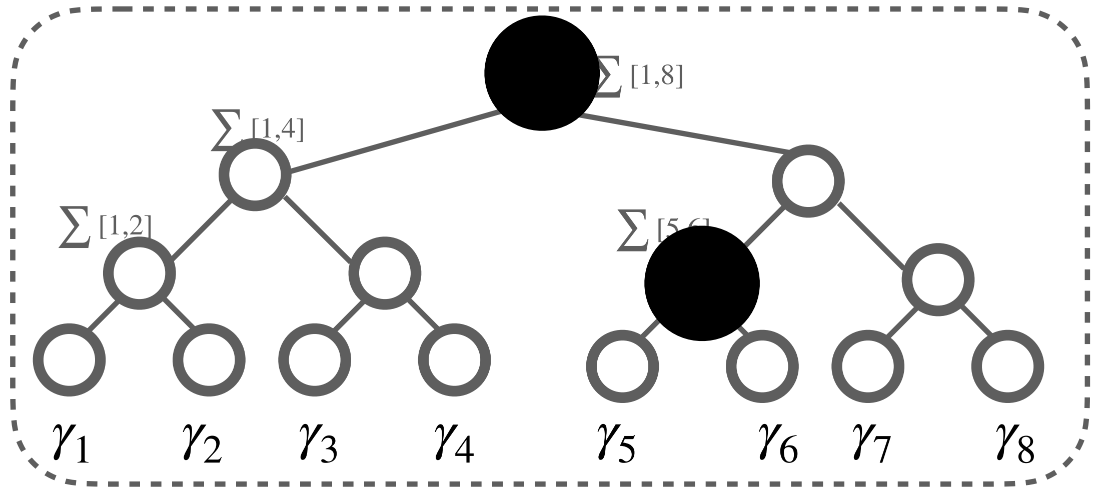


2. **Performance of P_{vec}**
   — it guarantees SDP for all $\epsilon \in (0,15)$, $\delta \in (0,1/2)$
   — the injected noise is $\frac{L^2}{\epsilon^2} \log^2(d/\delta)$ per entry **(indep. of n)**

(Essentially, it simulates central model without a trusted server)

Federated LCBs under SDP

Leverage vector-sum protocol



Can we simply use P_{vec} to add all p-sums across M agents?

The norm of p-sum could be linear with T !

— sum of M p-sums with P_{vec} (i.e., $n = M$)

— each data point has a large norm

View n in P_{vec} as data points across agents

—e.g., for $k = 6$

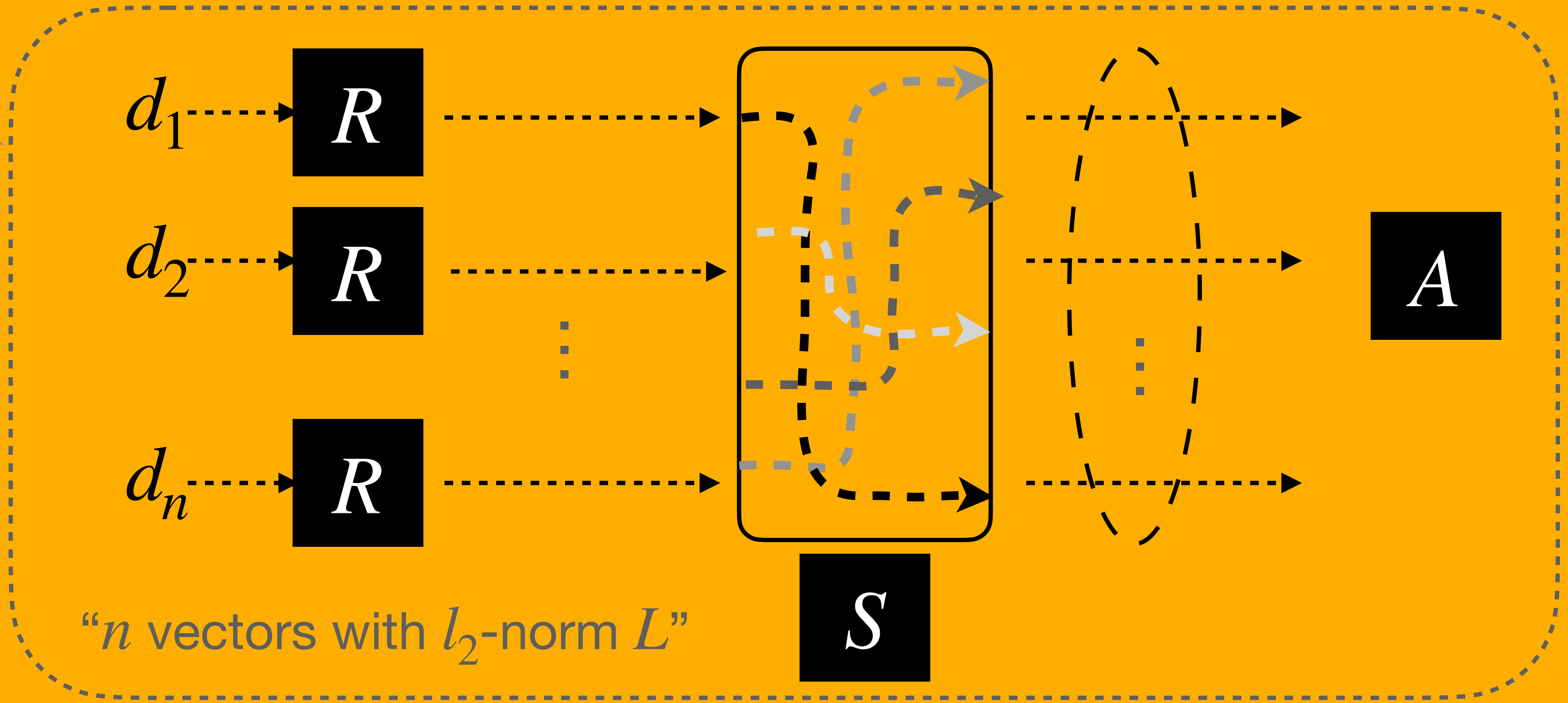
— each p-sum has $2B$ points

— $n = M \cdot 2B$ with each norm bounded

— each sync incurs only $1/\epsilon^2$ noise ✅

Differential Privacy 502

1. **How to achieve SDP?**
   — instead of using amplification lemma
   — one can use specific shuffle protocol
   — $P_{\text{vec}} = (R_{\text{vec}}, S, A_{\text{vec}})$[CJMP21] is one example



2. **Performance of P_{vec}**
   — it guarantees SDP for all $\epsilon \in (0,15)$, $\delta \in (0,1/2)$
   — the injected noise is $\frac{L^2}{\epsilon^2}\log^2(d/\delta)$ per entry **(indep. of n)**

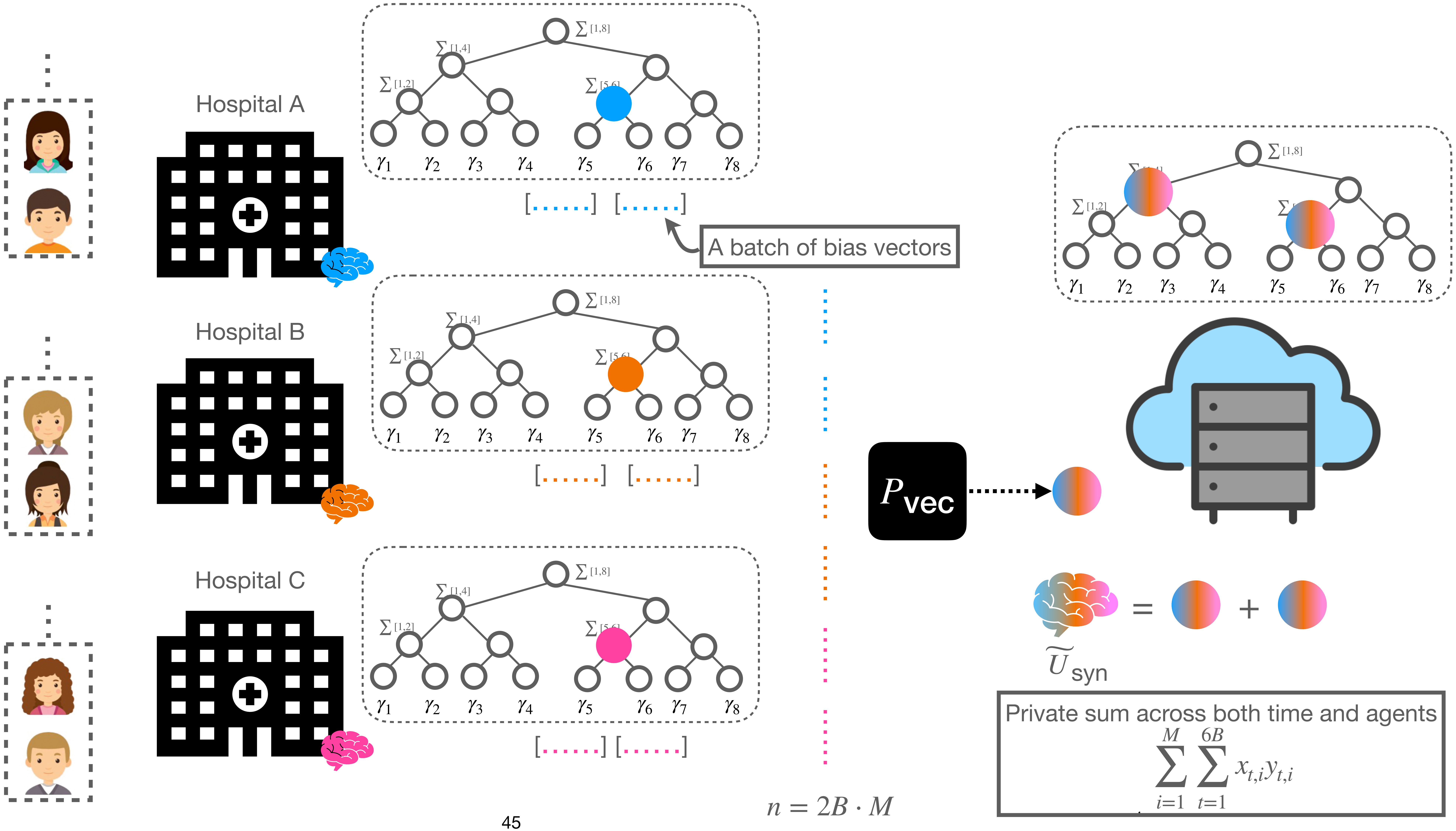
(Essentially, it simulates central model without a trusted server)

Algorithm in action

With P_{vec}

time $t = 6B$

(the 6-th communication)



Federated LCBs under SDP

Improved privacy via P_{vec}

Theorem 3 (Performance under SDP via P_{vec}, informal)

Let batch size $B = \sqrt{T/M}$ and $\kappa = 1 + \log(T/B)$. Combine P_{vec} with our privacy protocol. Then, `Private-FedLinUCB` enjoys

1. **Privacy** — (ϵ, δ)-SDP for any $\epsilon \in (0,60)$, $\delta \in (0,1)$,
2. **Regret** — $R_M(T) =$ non-private regret $+ \sqrt{MT}\dfrac{d^{3/4}\log^{3/4}(M\kappa/\delta)}{\sqrt{\epsilon}}$
3. **Communication** — $\sqrt{MT}$ rounds of sync between agents and server

Match the "lower bound"
Privacy cost is on the order of $\sqrt{MT}$

Privacy holds for a wide range of ϵ
This significantly improve upon the one via amplification

A more complicated algorithm
— need P_{vec}
— need store all data points

Analysis

A Generic Analysis

"One-line" proof for regret

Aggregated prefix sum
(sum over time and agents)

Privacy Noise Condition (PNC)

For any $t = kB$, let $N_{t,i}, n_{t,i}$ be total privacy noise injected in $\sum_{s=1}^{t} x_{s,i}x_{s,i}^\top$ and $\sum_{s=1}^{t} x_{s,i}y_{s,i}$, respectively

1. $\sum_{i \in [M]} n_{t,i}$ be a random vector, each entry is zero mean sub-Gaussian with variance at most σ_{tot}^2
2. $\sum_{i \in [M]} N_{t,i}$ be a random symmetric matrix, each entry is zero mean sub-Gaussian with variance at most σ_{tot}^2

Proposition 1 (Generic regret bound under PNC, informal)

Suppose that the privacy protocol satisfies PNC with parameter σ_{tot}^2, then `Private-FedLinUCB` enjoys the following regret with high probability

$$R_M(T) = \tilde{O}\left(dMB + d\sqrt{MT} + \sqrt{\sigma_{\text{tot}} MT} d^{3/4}\right)$$

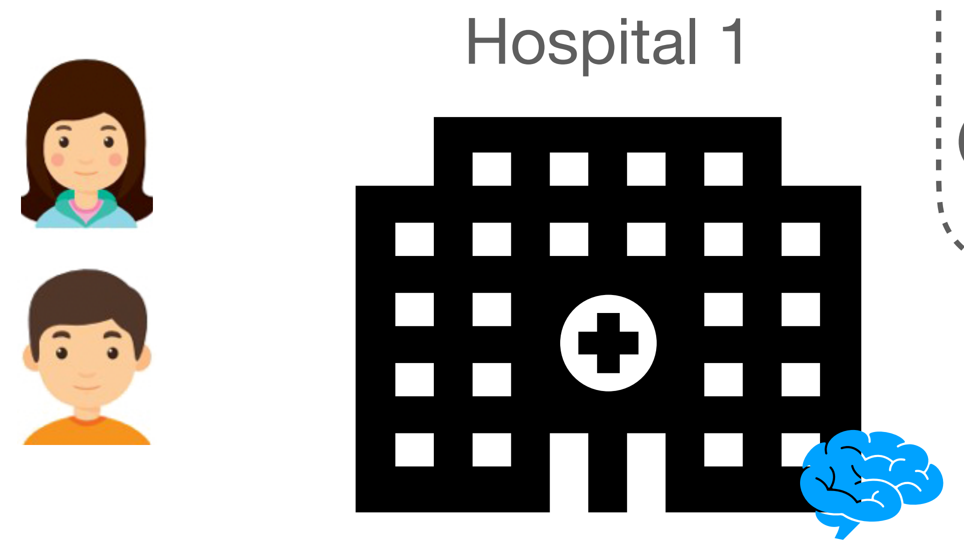
Cost due to batching | Standard regret | Cost due to privacy

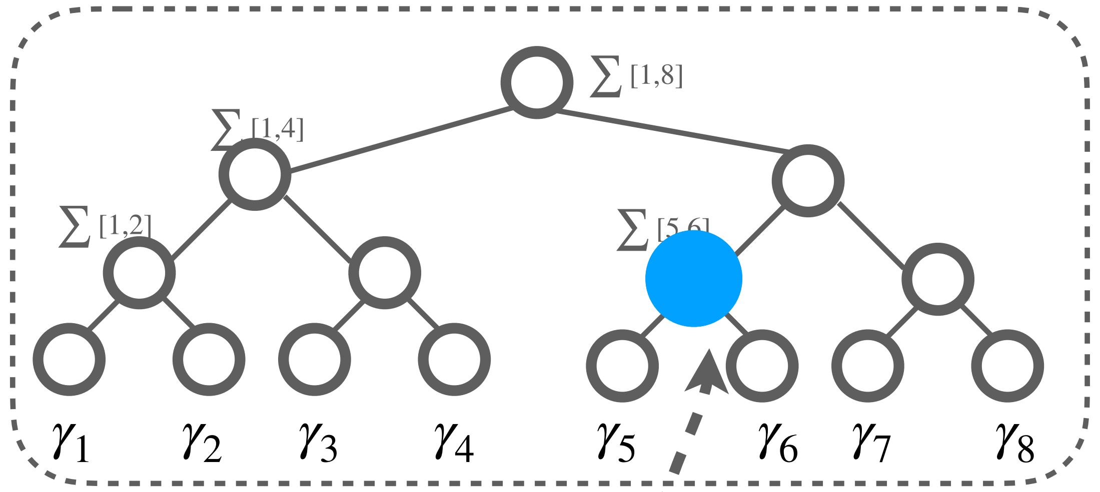
Total Privacy Noise

Silo-level LDP

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} M T}$

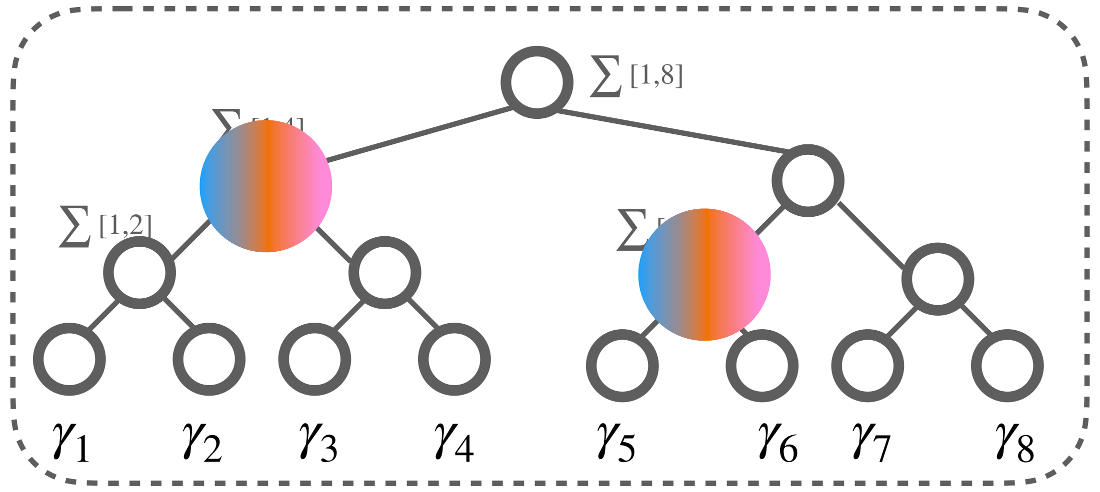
Regret under silo-level LDP: $\tilde{O}(M^{3/4}\sqrt{T/\epsilon})$





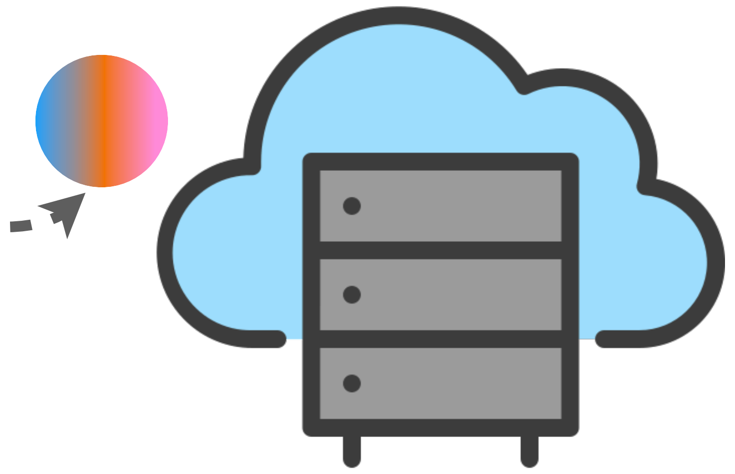
$\tilde{O}(1/\epsilon^2)$ privacy noise in **each** p-sum

(thanks to binary tree)



$\tilde{O}(M/\epsilon^2)$ privacy noise in **aggregated** p-sum

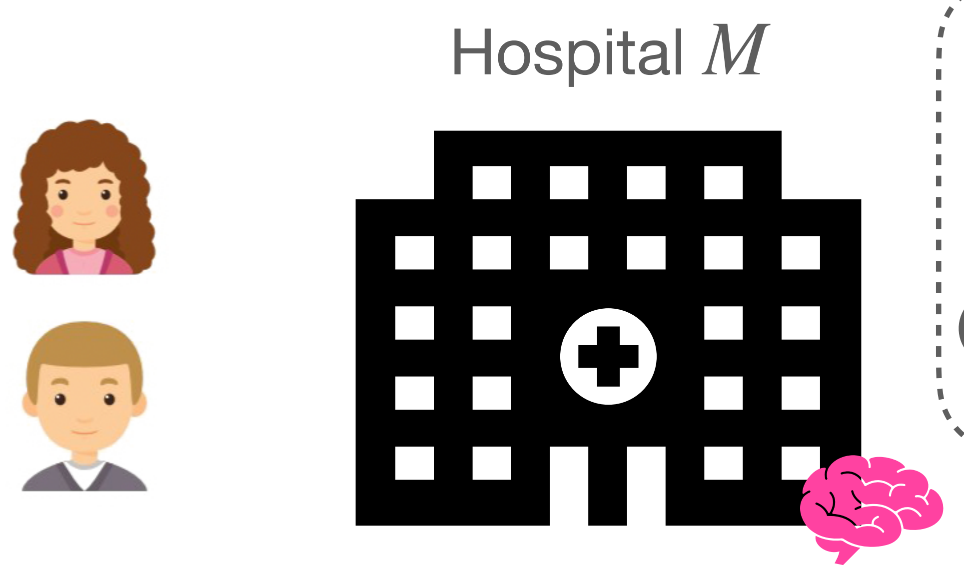
(sum of M noise)

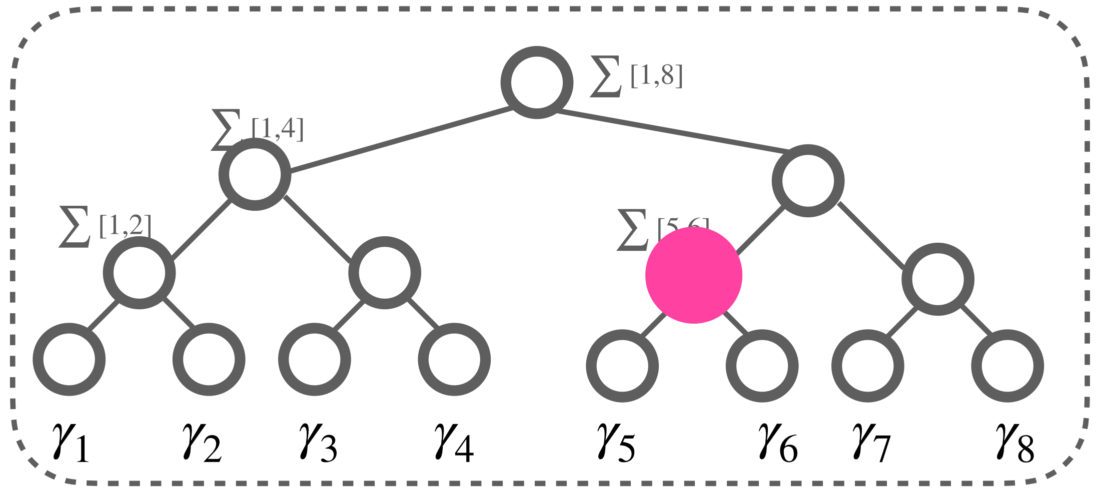


$\tilde{O}(M/\epsilon^2)$ privacy noise in **aggregated prefix** sum

i.e., σ_{tot}^2

(sum of $\log K$ noise)





Private sum across both time and agents

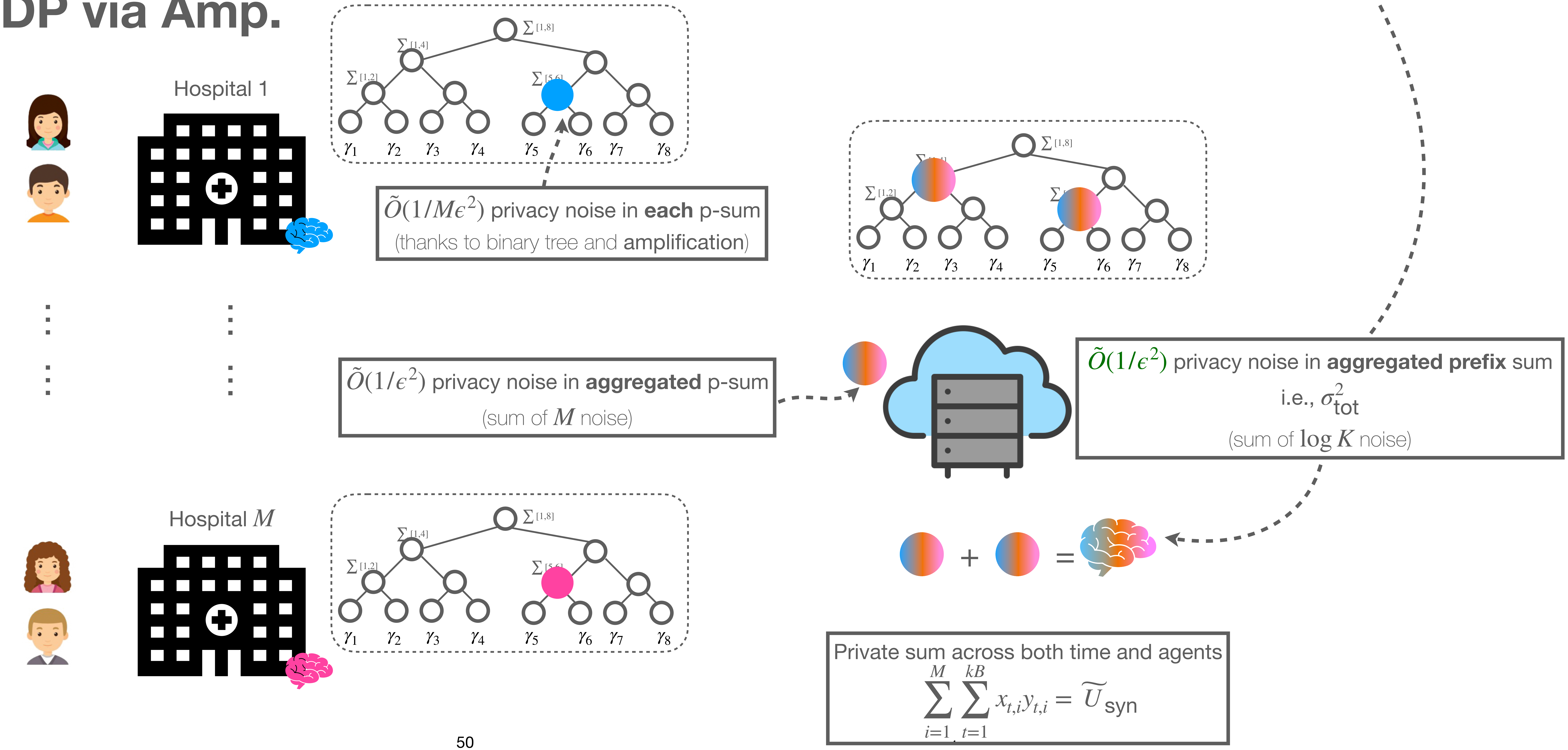
$$\sum_{i=1}^{M}\sum_{t=1}^{kB} x_{t,i} y_{t,i} = \widetilde{U}_{\text{syn}}$$

Total Privacy Noise

SDP via Amp.

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} MT}$

Regret under SDP: $\tilde{O}(\sqrt{MT/\epsilon})$

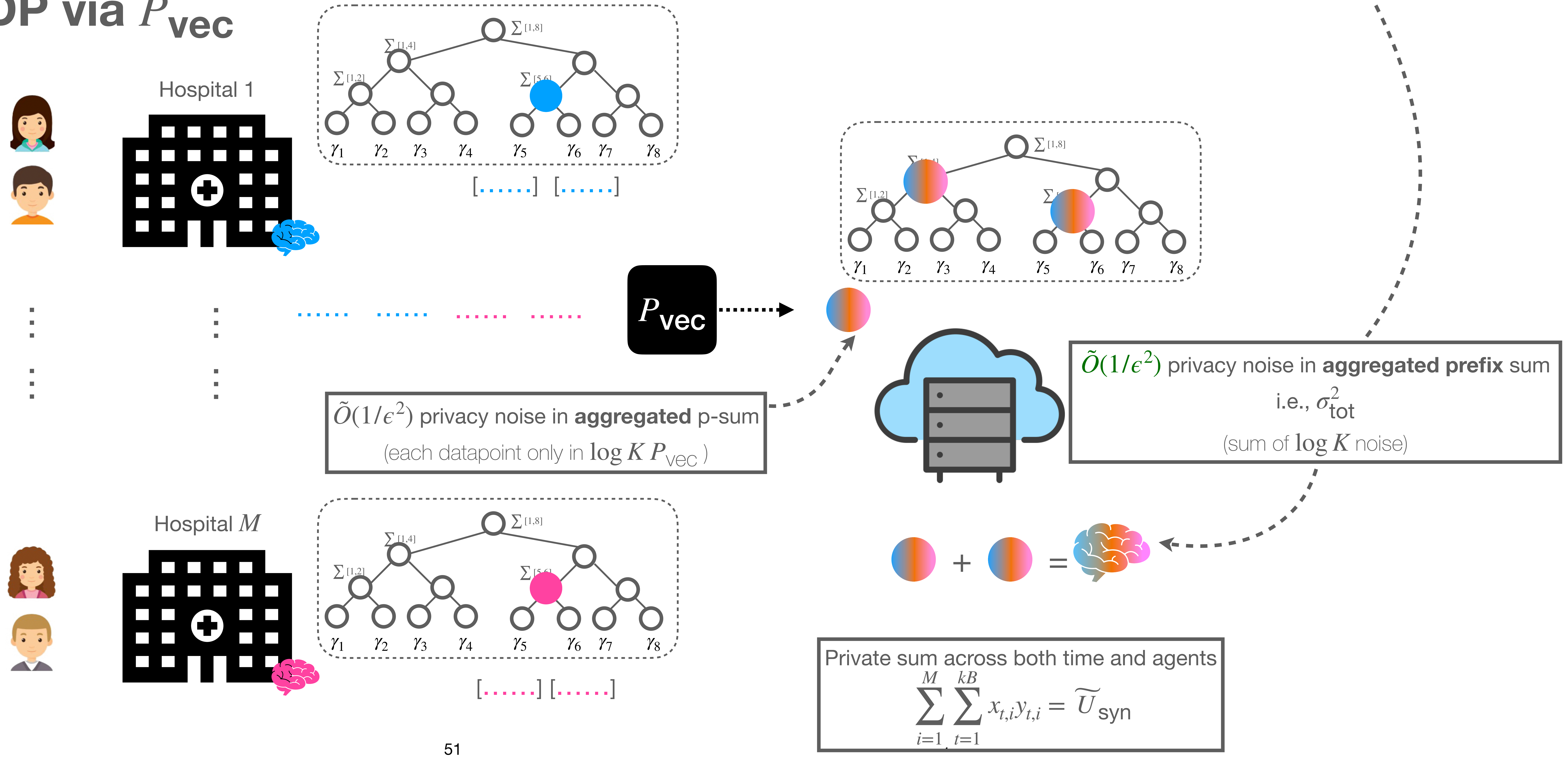


Total Privacy Noise

SDP via P_{vec}

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} MT}$

Regret under SDP: $\tilde{O}(\sqrt{MT/\epsilon})$



Importance of p-sum

Why P_{alt} fails for SDP

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} MT}$

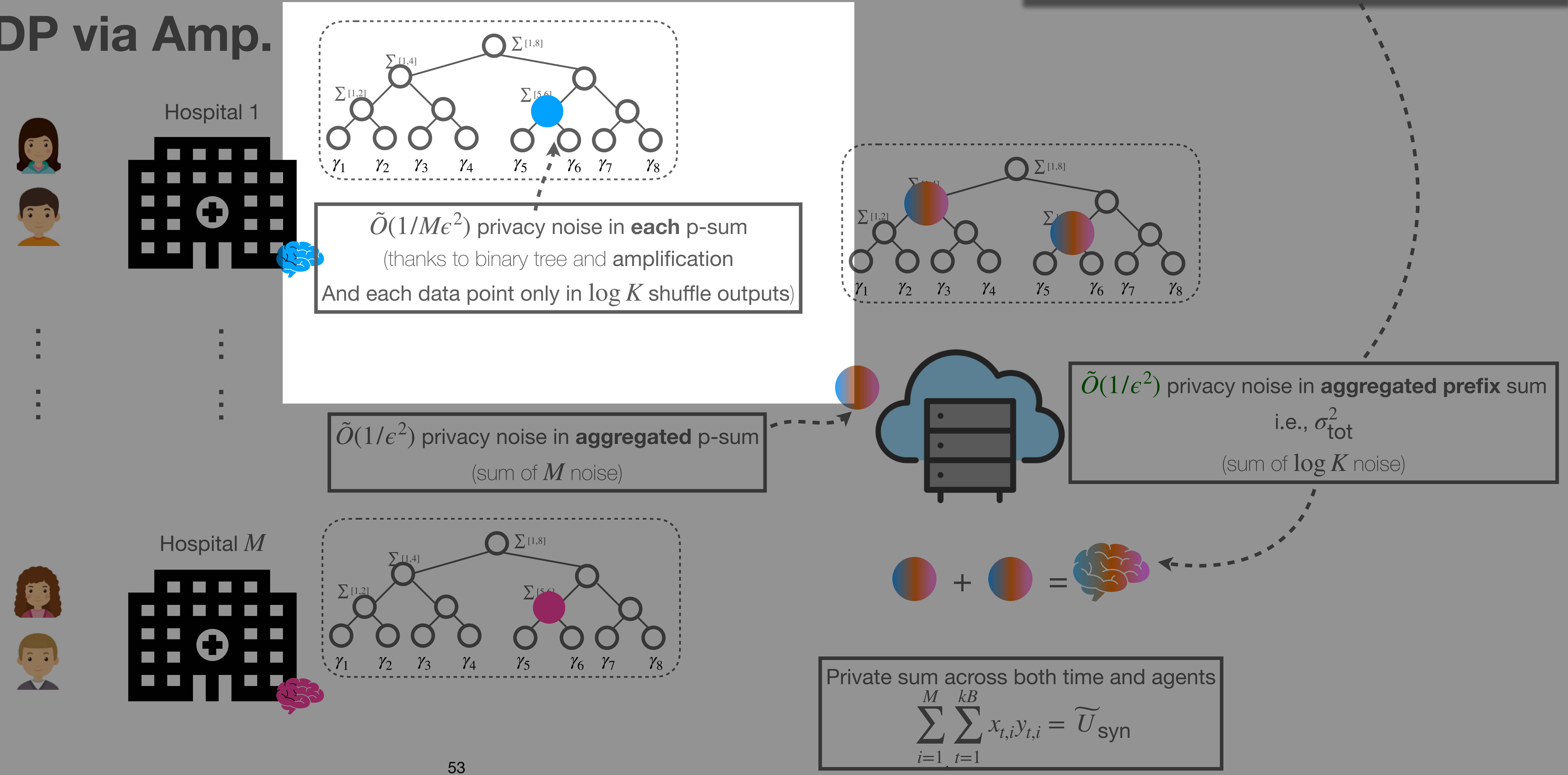
Regret under SDP: $\tilde{O}(\sqrt{MT/\epsilon})$

Importance of p-sum

SDP via Amp.

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} MT}$

Regret under SDP: $\tilde{O}(\sqrt{MT/\epsilon})$

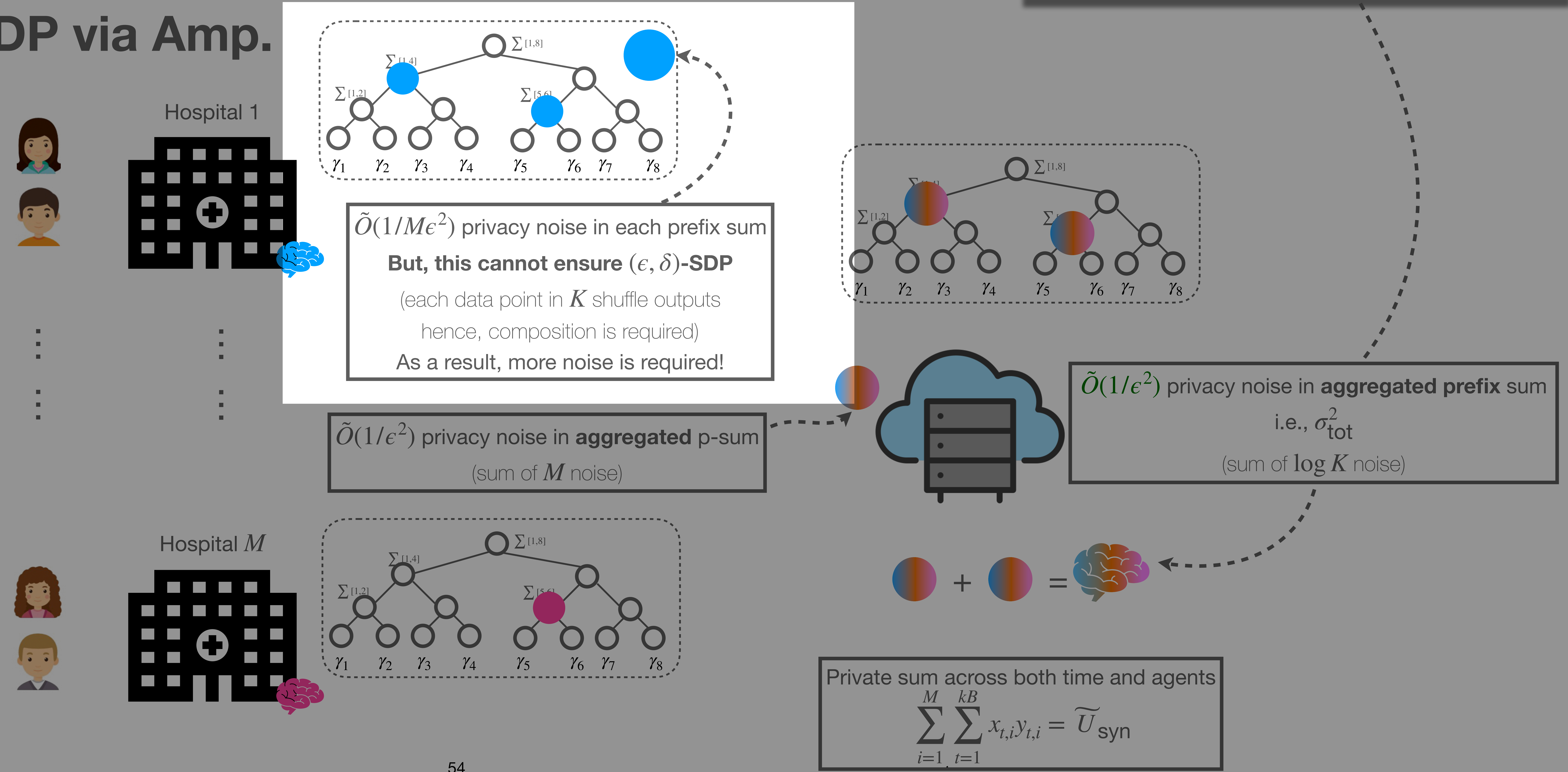


Importance of p-sum

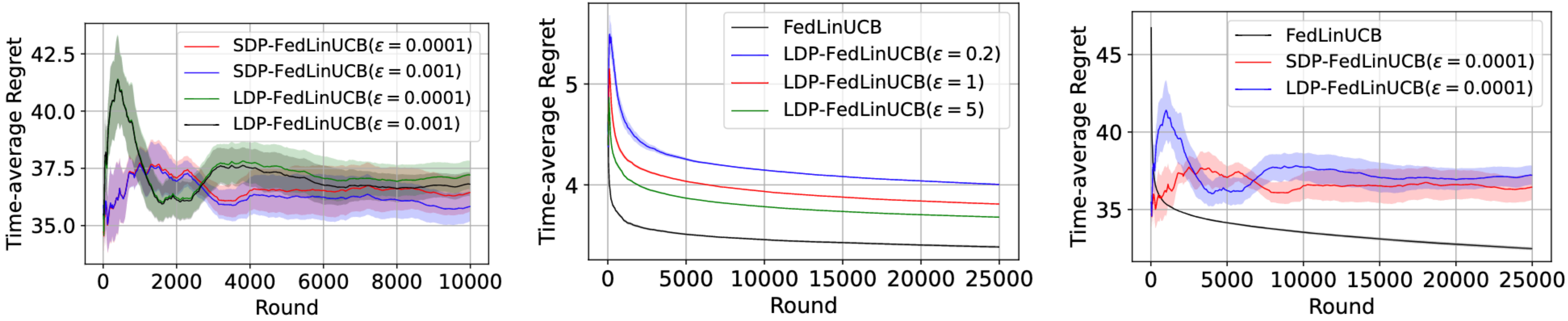
SDP via Amp.

Prop. 1. Regret due to privacy: $\sqrt{\sigma_{\text{tot}} MT}$

Regret under SDP: $\tilde{O}(\sqrt{MT/\epsilon})$



Simulations



(a) Synthetic data ($M = 100$) (b) Real data ($M = 10$) (c) Real data ($M = 100$)

Discussions

Q1: Can we further reduce comm. cost to $\log T$

Then, it might need adaptive update based on determinant condition. Challenges exist in private case

Q2: Silo-level LDP/ SDP vs. other privacy notions in contextual bandits?

We give a comprehensive discussions on difference and connections

Q3: What if users even do not trust each local agent?

It turns out that a simple tweak of our algorithm can handle this situation

Q4: What if users participate multiple times ? (within one silo or across silos)

One can use composition or group privacy to handle. Or directly analyze the total sensitivity

Q5: How to balance between privacy and algorithm complexity?

Good question. We are working on it right now

Q6: Can we generalize it to federated RL

Yes, at least for RL with linear function approximation

One last thing…

Recent Research…

Private Online Learning

- **Private MAB**
  - — “MAB under local DP with tight lower bound” [RZLS20, arxiv]
  - — “the state-of-the-art of private MAB for all three DP models” [CZ*23, ICLR23]
  - — “private and robust MAB” [WZ*TW23, submitted]
- **Private Contextual Bandits**
  - — “linear contextual bandits under shuffle model” [CZ*22, ICML22]
  - — “federated LCBs under both silo-level LDP and SDP [ZC, arixv, submitted]
  - — “kernel bandits under local model” [ZT21, AAAI21]
  - — “private linear bandits with distributed feedback” [LZJ22, WiOpt22, Best Student Paper]
  - — “private distributed kernel bandits” [LZJ23, Sigmetrics23]
- **Private RL**
  - — “A comprehensive study of tabular RL under both central and local DP models” [CZ*22, AAAI22, oral]
  - — “Private heavy-tailed RL”[WZ*CW23, ICML23]
  - — “The first study of private RL with linear function approximation”[Z22, Sigmetrics22]
  - — “Study of private LQR” [CZ*S21, ISIT21]

Many interesting open problems in this area…
Collaborations are welcome 🎉

[Z* means co-primary authors]

Recent Research…

Private Online Learning

- **Private MAB**
  - “MAB under local DP with tight lower bound” [RZLS20, arxiv]
  - “the state-of-the-art of private MAB for all three DP models” [CZ*23, ICLR23]
  - “private and robust MAB” [WZ*TW23, submitted]
- **Private Contextual Bandits**
  - “linear contextual bandits under shuffle model” [CZ*22, ICML22]
  - “federated LCBs under both silo-level LDP and SDP [ZC, arixv, submitted]
  - “kernel bandits under local model” [ZT21, AAAI21]
  - “private linear bandits with distributed feedback” [LZJ22, WiOpt22, Best Student Paper]
  - “private distributed kernel bandits” [LZJ23, Sigmetrics23]
- **Private RL**
  - “A comprehensive study of tabular RL under both central and local DP models” [CZ*22, AAAI22, oral]
  - “Private heavy-tailed RL”[WZ*CW23, ICML23]
  - “The first study of private RL with linear function approximation”[Z22, Sigmetrics22]
  - “Study of private LQR” [CZ*S21, ISIT21]

Many interesting open problems in this area…
Collaborations are welcome 🎉

[Z* means co-primary authors]

Recent Research…

Bayesian Optimization (BO) and RL

How to make BO and RL more practical?

heavy-tailed rewards, non-stationarity, constraints

NeurIPS'22

On Kernelized Multi-Armed Bandits with Constraints

Xingyu Zhou
Electrical and Computer Engineering
Wayne State University
Detroit, MI, USA
xingyu.zhou@wayne.edu

Bo Ji
Computer Science
Virginia Tech
Blacksburg, VA, USA
boji@vt.edu

AISTATS'22

Weighted Gaussian Process Bandits for Non-stationary Environments

Yuntian Deng
The Ohio State University

Xingyu Zhou
Wayne State University

Baekjin Kim
University of Michigan

Ambuj Tewari
University of Michigan

Abhishek Gupta
The Ohio State University

Ness Shroff
The Ohio State University

NeurIPS'22

Provably Efficient Model-Free Constrained RL with Linear Function Approximation

Arnob Ghosh
Electrical and Computer Engineering
The Ohio State University
Columbus, OH, USA
ghosh.244@osu.edu

Xingyu Zhou
Electrical and Computer Engineering
Wayne State University
Detroit, MI, USA
xingyu.zhou@wayne.edu

Ness Shroff
Electrical and Computer Engineering
The Ohio State University
Columbus, OH, USA
shroff.11@osu.edu

ICML'23

Differentially Private Episodic Reinforcement Learning with Heavy-tailed Rewards

Yulian Wu [1 2] **Xingyu Zhou** [3] **Sayak Ray Chowdhury** [4] **Di Wang** [1 2]

ICLR'23

ACHIEVING SUB-LINEAR REGRET IN INFINITE HORIZON AVERAGE REWARD CONSTRAINED MDP WITH LINEAR FUNCTION APPROXIMATION

Arnob Ghosh
Electrical and Computer Engineering
The Ohio State University
ghosh.244@osu.edu

Xingyu Zhou
Electrical and Computer Engineering
Wayne State University
xingyu.zhou@wayne.edu

Ness Shroff
Electrical and Computer Engineering
The Ohio State University
shroff.11@osu.edu

Sigmetrics'23

(Private) Kernelized Bandits with Distributed Biased Feedback

FENGJIAO LI, Virginia Tech, USA
XINGYU ZHOU, Wayne State University, USA
BO JI, Virginia Tech, USA

Reference

- **[KMA+19]** Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. Advances and open problems in federated learning. arXiv preprint:1912.04977, 2019
- **[APS11]** Abbasi-Yadkori, Yasin, Dávid Pál, and Csaba Szepesvári. "Improved algorithms for linear stochastic bandits." *NeurIPS,* 2011.
- **[FJR15]** Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pages 1322–1333, 2015.
- **[HZL19]** Zecheng He, Tianwei Zhang, and Ruby B Lee. Model inversion attacks against collaborative inference. In Proceedings of the 35th Annual Computer Security Applications Conference, pages 148–162, 2019
- **[SSS+17]** Shokri R, Stronati M, Song C, Shmatikov V. Membership inference attacks against machine learning models. In2017 IEEE symposium on security and privacy (SP) 2017 May 22 (pp. 3-18). IEEE.
- **[DR14]** Dwork, Cynthia, and Aaron Roth. "The algorithmic foundations of differential privacy." *Foundations and Trends® in Theoretical Computer Science* 9.3–4 (2014): 211-407.
- **[LR21]** Private federated learning without a trusted server: Optimal algorithms for convex losses. *arXiv preprint arXiv:2106.09779*, 2021
- **[LHW+22]** On privacy and personalization in cross-silo federated learning. *arXiv preprint arXiv:2206.07902*, 2022
- **[CSS11]** Chan, T-H. Hubert, Elaine Shi, and Dawn Song. "Private and continual release of statistics." *ACM Transactions on Information and System Security (TISSEC)* 14.3 (2011): 1-24.
- **[CSUZZ19]** Cheu, A., Smith, A., Ullman, J., Zeber, D., & Zhilyaev, M. (2019). Distributed differential privacy via shuffling. In Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part I 38 (pp. 375-403). Springer International Publishing.
- **[FMT20]** Feldman, Vitaly, Audra McMillan, and Kunal Talwar. "Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling." In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 954-964. IEEE, 2022
- **[CJMP21]** Cheu, Albert, Matthew Joseph, Jieming Mao, and Binghui Peng. "Shuffle private stochastic convex optimization." *arXiv preprint arXiv:2106.09805* (2021).
- **[RZLS20]** Ren, Wenbo, Xingyu Zhou, Jia Liu, and Ness B. Shroff. "Multi-armed bandits with local differential privacy." *arXiv preprint arXiv:2007.03121* (2020).
- **[CZ*23]** Chowdhury, Sayak Ray, and Xingyu Zhou. "Distributed Differential Privacy in Multi-Armed Bandits." ICLR23.
- **[WZ*TW]** Wu, Yulian, Xingyu Zhou, Youming Tao, and Di Wang. "On Private and Robust Bandits." *arXiv preprint arXiv:2302.02526* (2023).
- **[CZ*22]** Chowdhury, Sayak Ray, and Xingyu Zhou. "Shuffle private linear contextual bandits." *ICML22*.
- **[ZC23]** Xingyu Zhou, and Chowdhury, Sayak Ray "On Differentially Private Federated Linear Contextual Bandits"
- **[ZT21]** Zhou, Xingyu, and Jian Tan. "Local differential privacy for bayesian optimization." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 11152-11159. 2021.

Reference

- **[LZJ22]** Li, Fengjiao, Xingyu Zhou, and Bo Ji. "Differentially private linear bandits with partial distributed feedback." In *2022 20th International Symposium on Modeling and Optimization in Mobile, Ad hoc, and Wireless Networks (WiOpt)*, pp. 41-48. IEEE, 2022.
- **[LZJ23]** Li, Fengjiao, Xingyu Zhou, and Bo Ji. "(Private) Kernelized Bandits with Distributed Biased Feedback." *arXiv preprint arXiv:2301.12061* (2023).
- **[CZ*22]** Chowdhury, Sayak Ray, and Xingyu Zhou. "Differentially private regret minimization in episodic markov decision processes." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 6, pp. 6375-6383. 2022.
- **[Z22]** Zhou, Xingyu. "Differentially private reinforcement learning with linear function approximation." *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, no. 1 (2022): 1-27.
- **[CZ*]** Chowdhury, Sayak Ray, Xingyu Zhou, and Ness Shroff. "Adaptive control of differentially private linear quadratic systems." In *2021 IEEE International Symposium on Information Theory (ISIT)*, pp. 485-490. IEEE, 2021.

Thank you!