Towards *Private* and *Efficient* Cross-Device Federated Learning

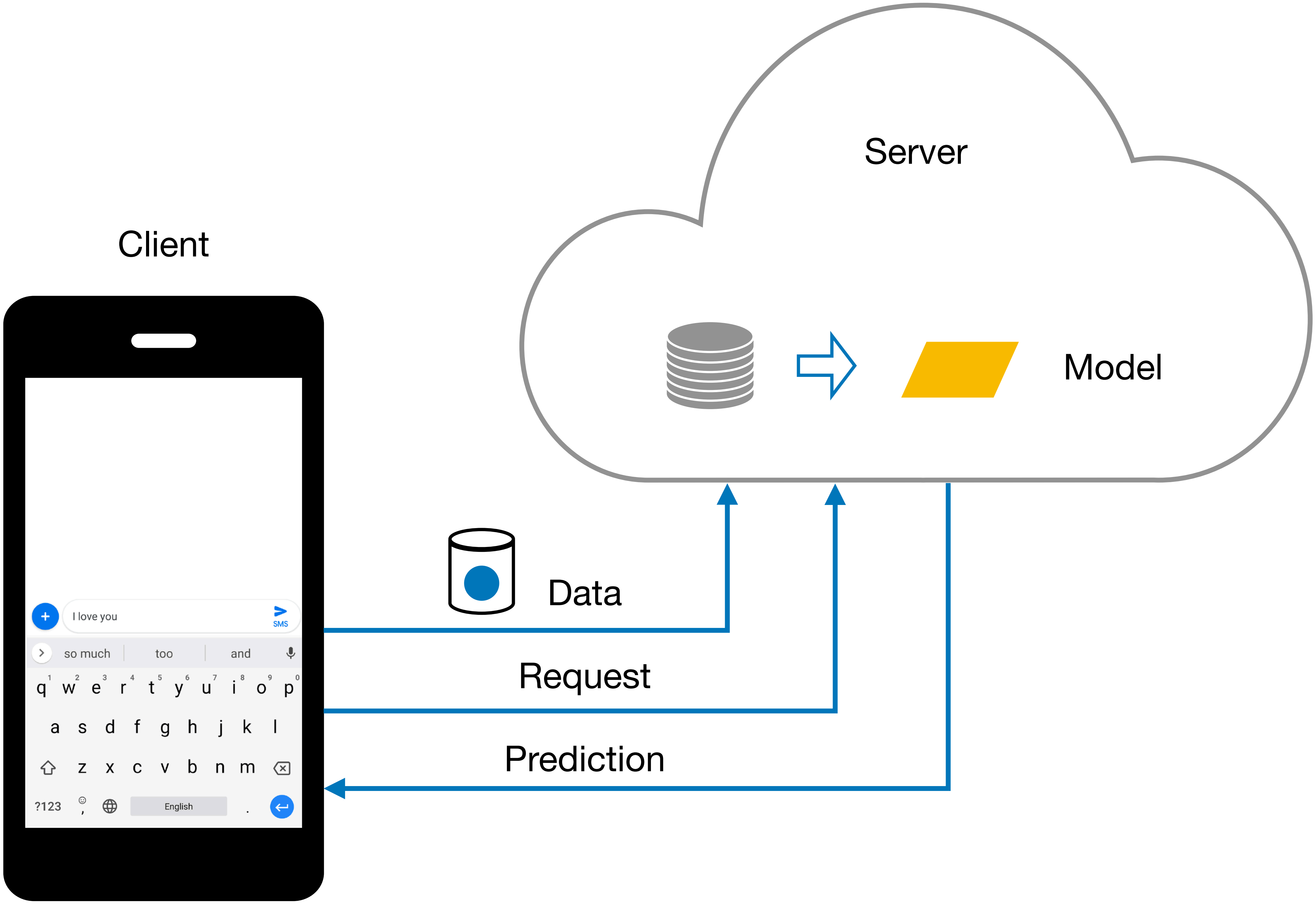
Zhifeng Jiang

Ph.D. Thesis Proposal Defense
Advisor: Wei Wang
Chairperson: Shuai Wang
Committee: Bo Li, Yangqiu Song

香港科技大學
THE HONG KONG UNIVERSITY OF
SCIENCE AND TECHNOLOGY

Apr 8, 2024

Centralized learning

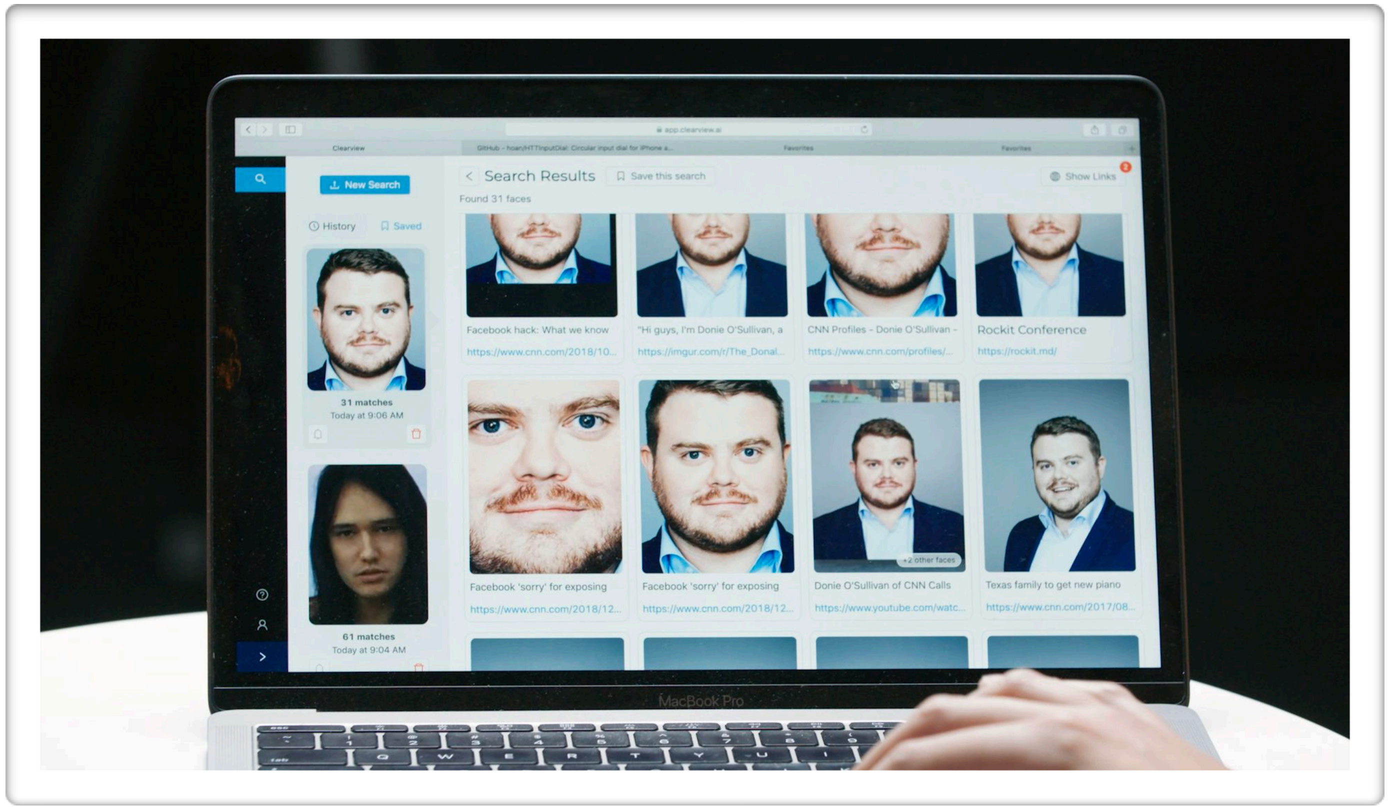


Centralized learning hurts privacy

Data breaches…

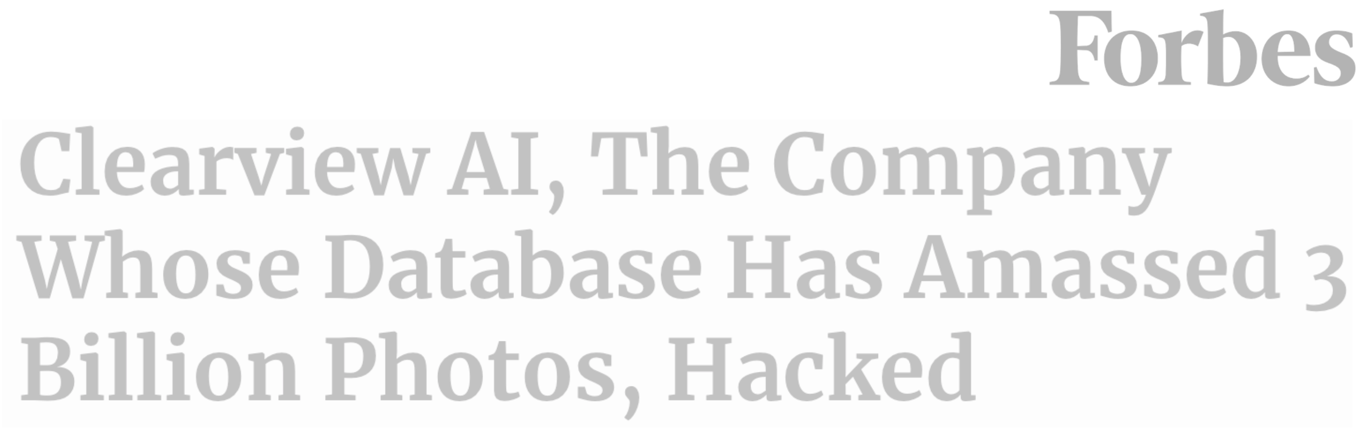
Forbes

Clearview AI, The Company Whose Database Has Amassed 3 Billion Photos, Hacked


New Search
History
Saved
Search Results
Save this search
Show Links
Found 31 faces
Facebook hack: What we know
"Hi guys, I'm Donie O'Sullivan, a
CNN Profiles - Donie O'Sullivan -
Rockit Conference
31 matches
Today at 9:06 AM
61 matches
Today at 9:04 AM
Facebook 'sorry' for exposing
Facebook 'sorry' for exposing
Donie O'Sullivan of CNN Calls
Texas family to get new piano
MacBook Pro

Centralized learning hurts privacy

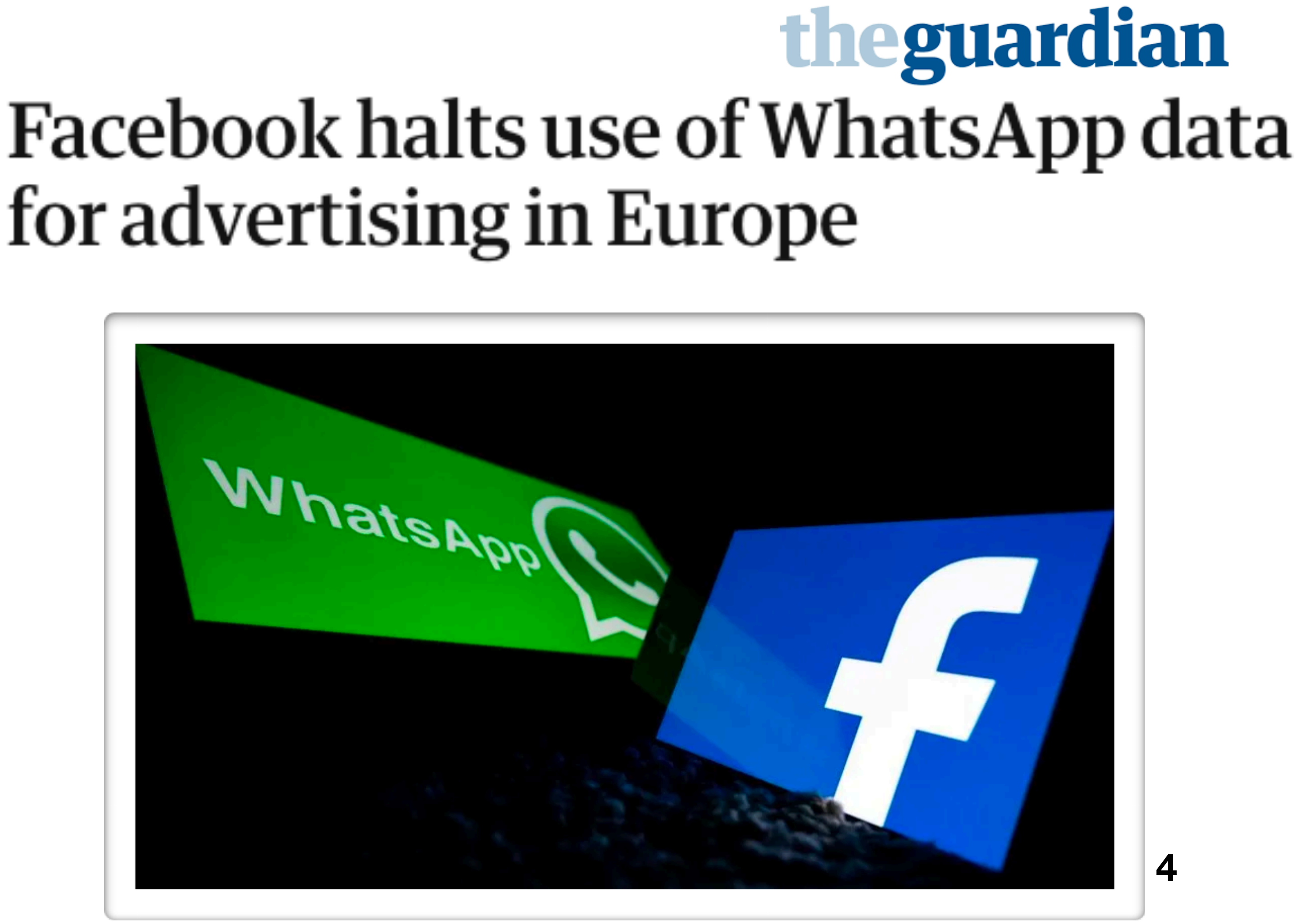
Data breaches…



Forbes

Clearview AI, The Company Whose Database Has Amassed 3 Billion Photos, Hacked

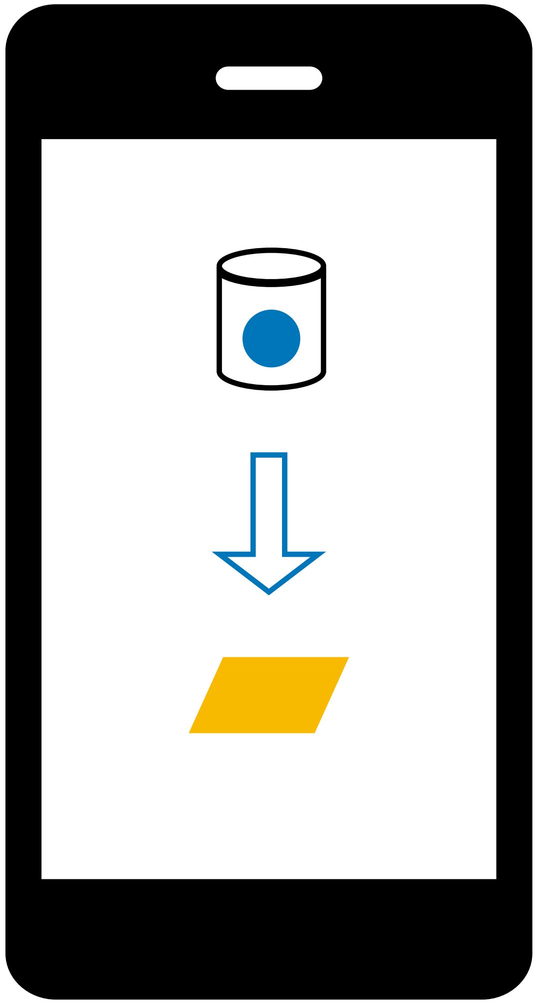
Potential abuse…



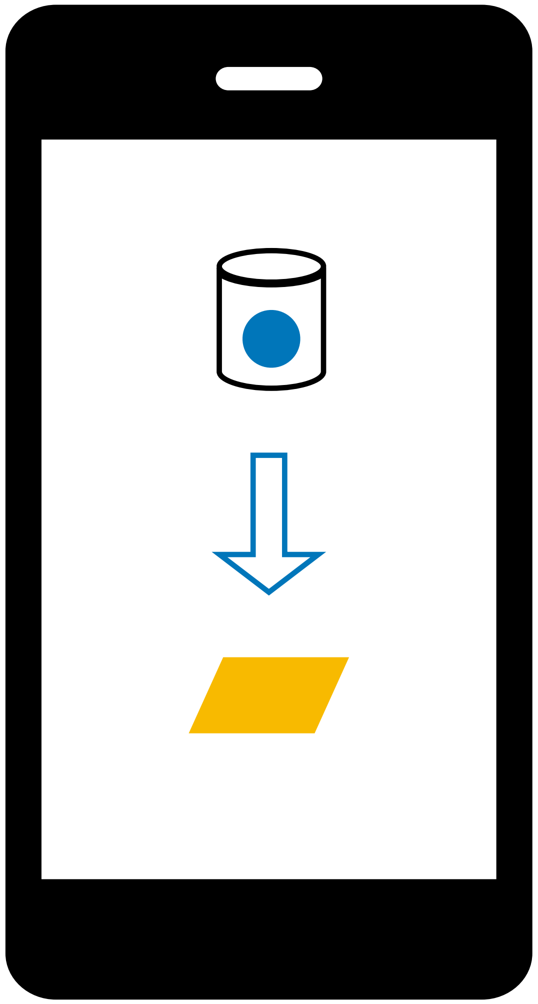
theguardian

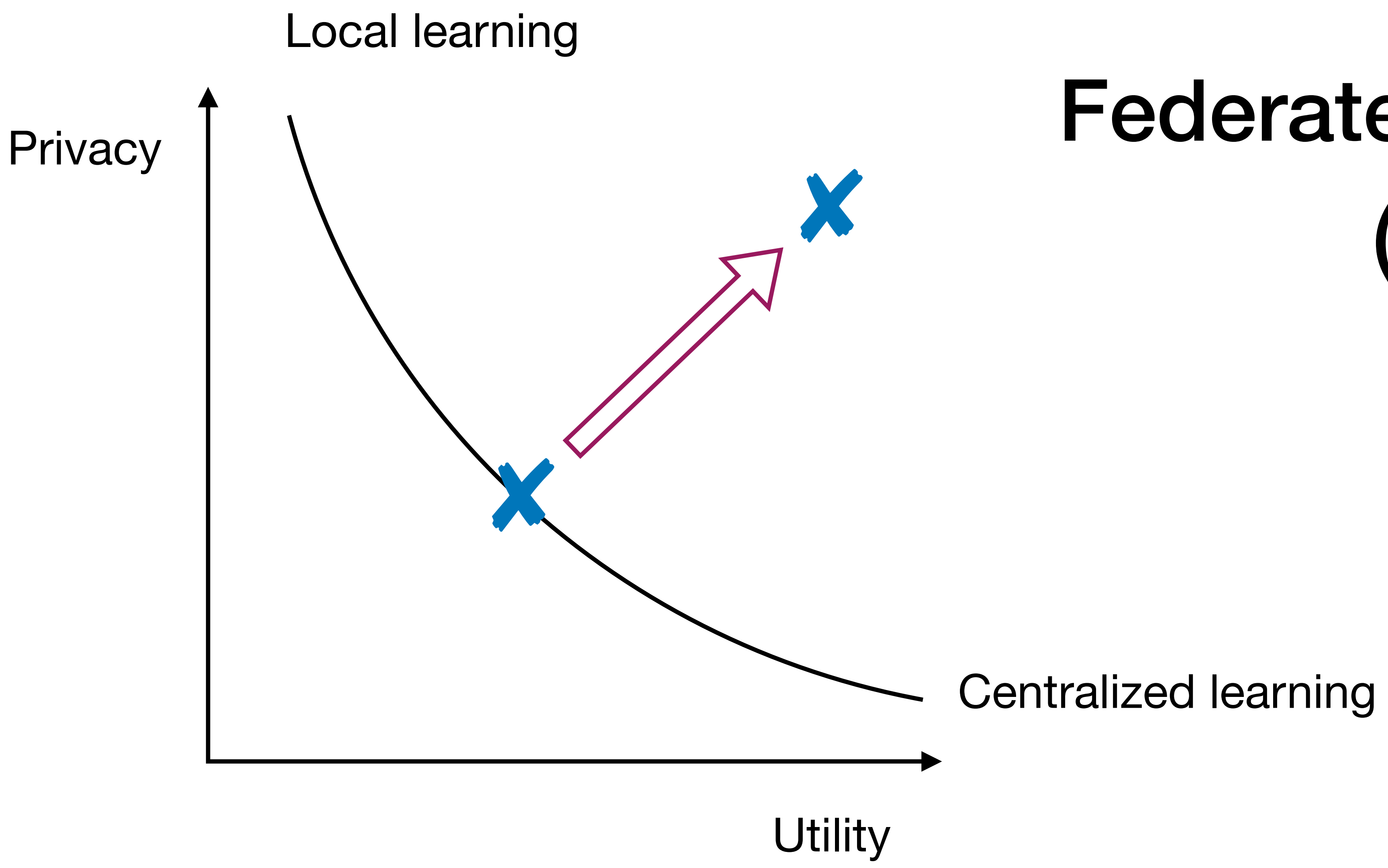
Facebook halts use of WhatsApp data for advertising in Europe

Local learning



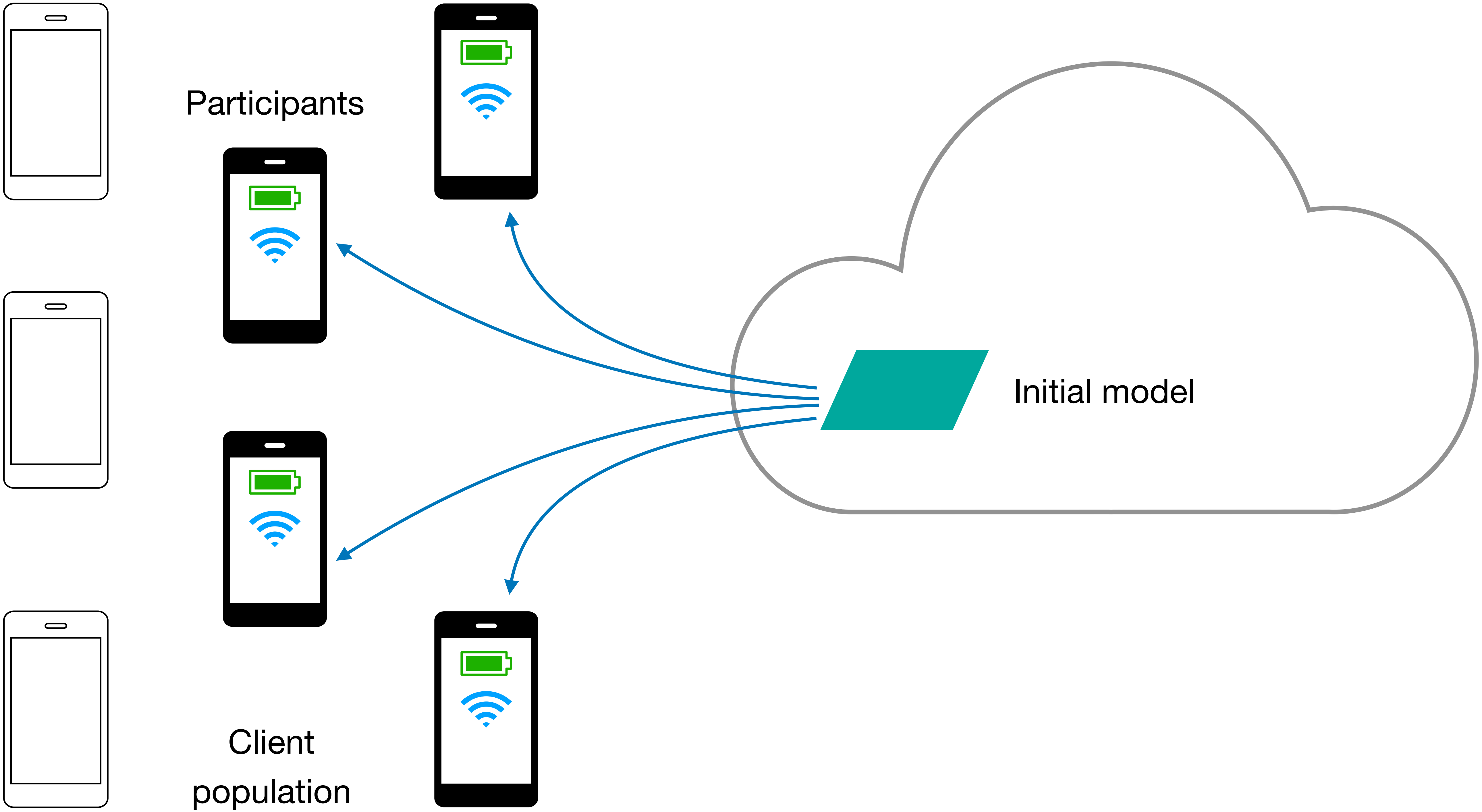
Local learning suffers from low data quality



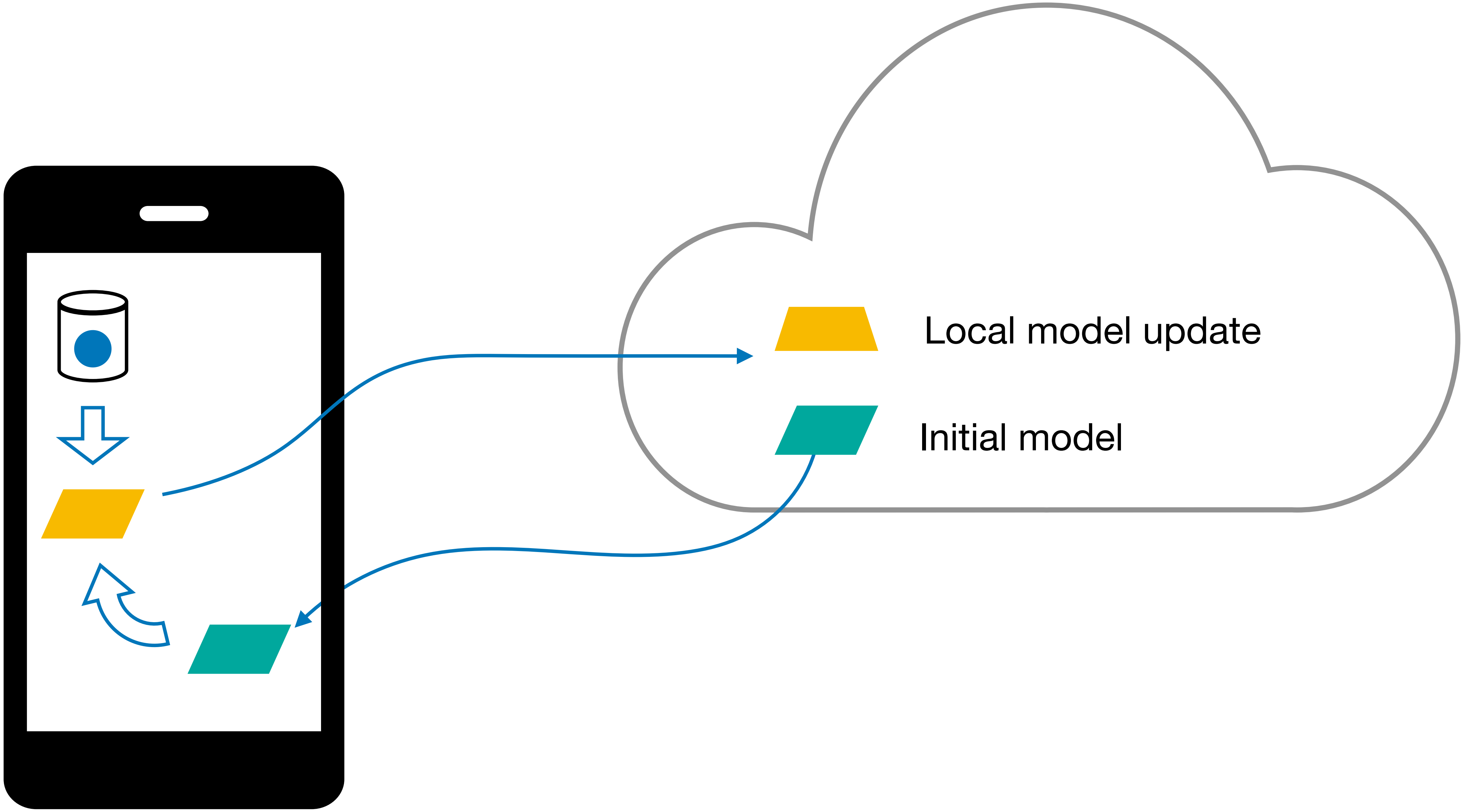


Federated Learning (FL)

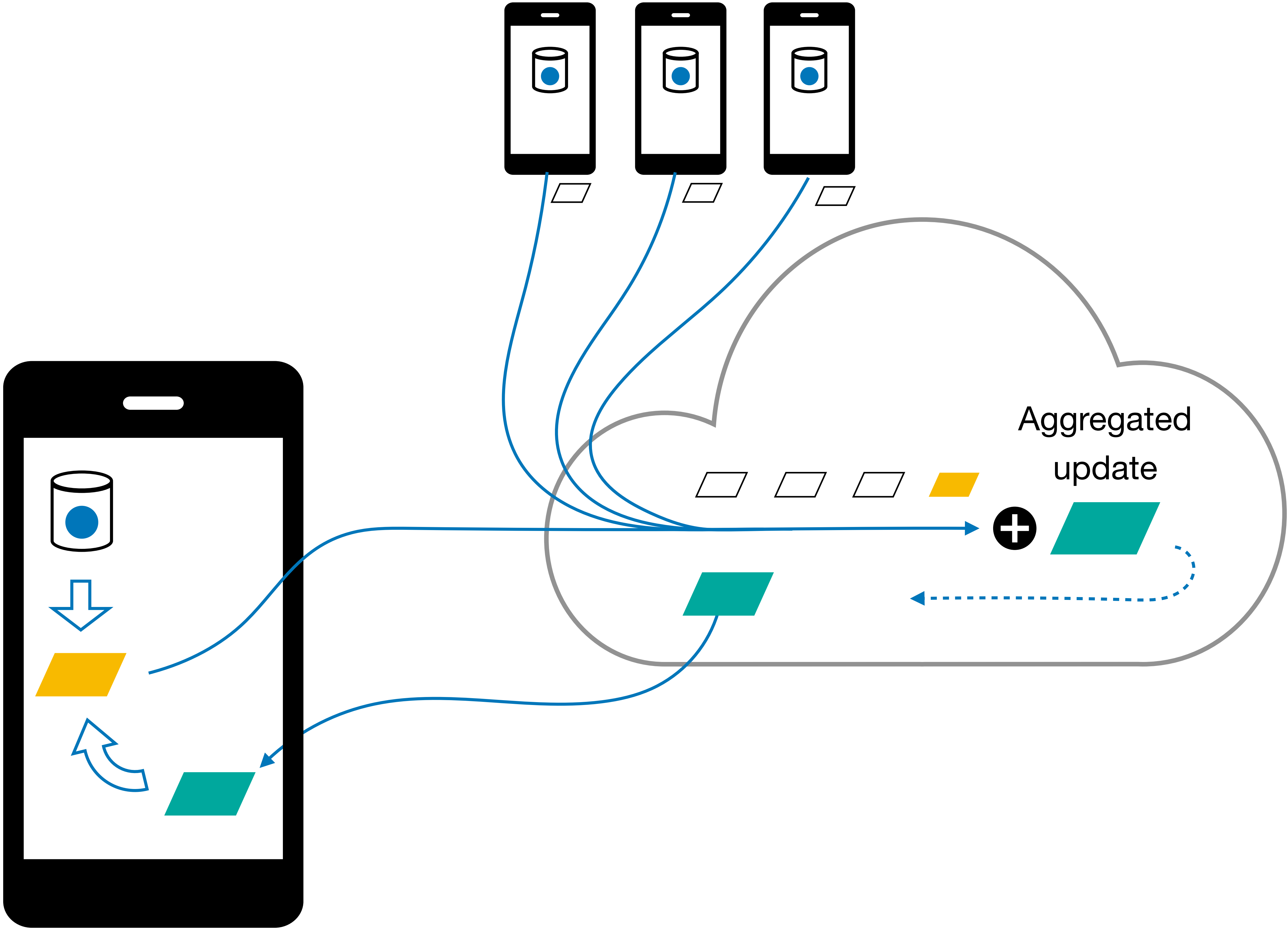
Step 1: Participant Selection



Step 2: Local Training

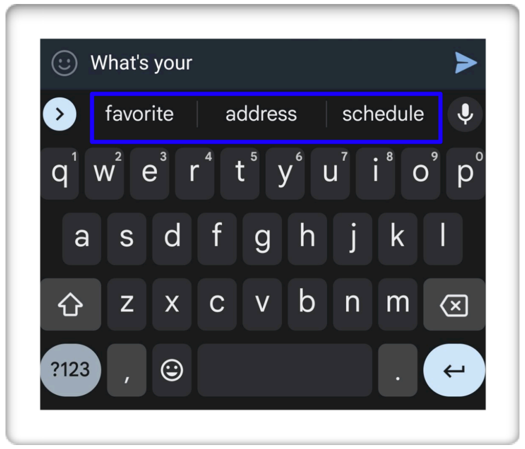


Step 3: Model Aggregation



Cross-Device Applications

Mobile



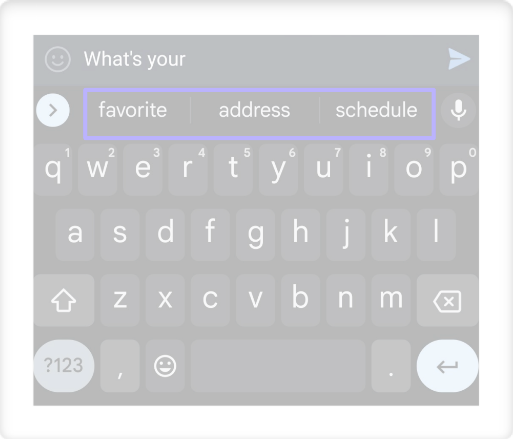
What's your

favorite | address | schedule

Google's Keyboard

Cross-Device Applications

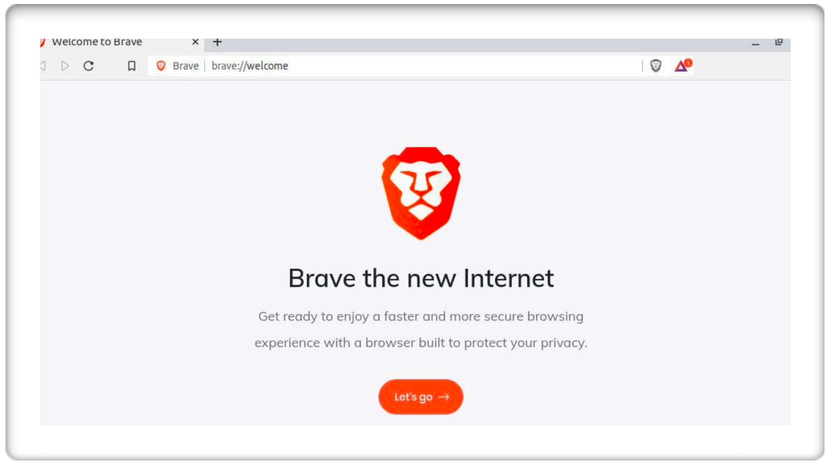
Mobile



Google's Keyboard

Apple's speaker recognition

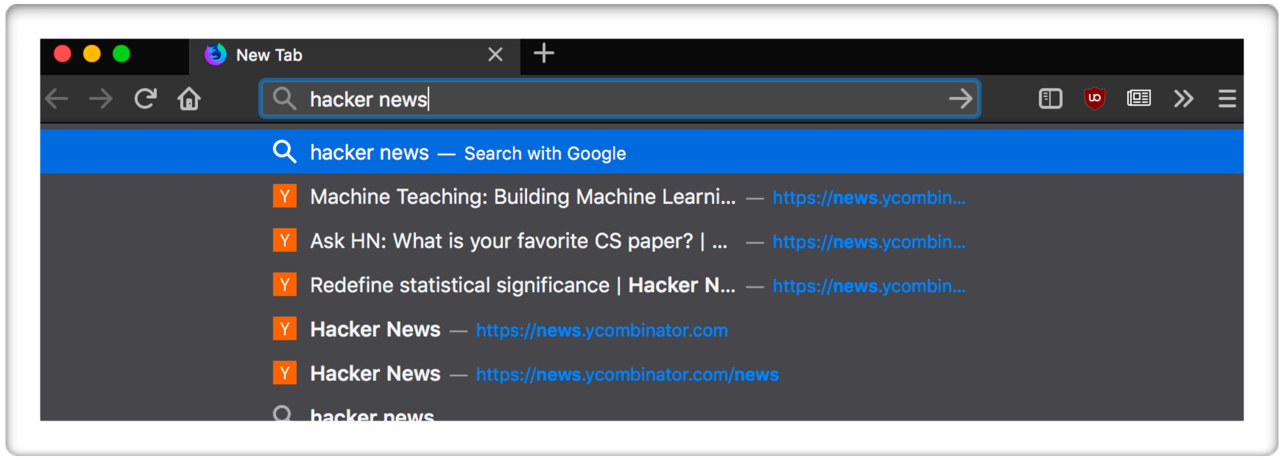
Huawei's ads recommendation



Brave the new Internet

Get ready to enjoy a faster and more secure browsing experience with a browser built to protect your privacy.

Brave's news recommendation



hacker news

hacker news — Search with Google

Machine Teaching: Building Machine Learni... — https://news.ycombin...

Ask HN: What is your favorite CS paper? | ... — https://news.ycombin...

Redefine statistical significance | Hacker N... — https://news.ycombin...

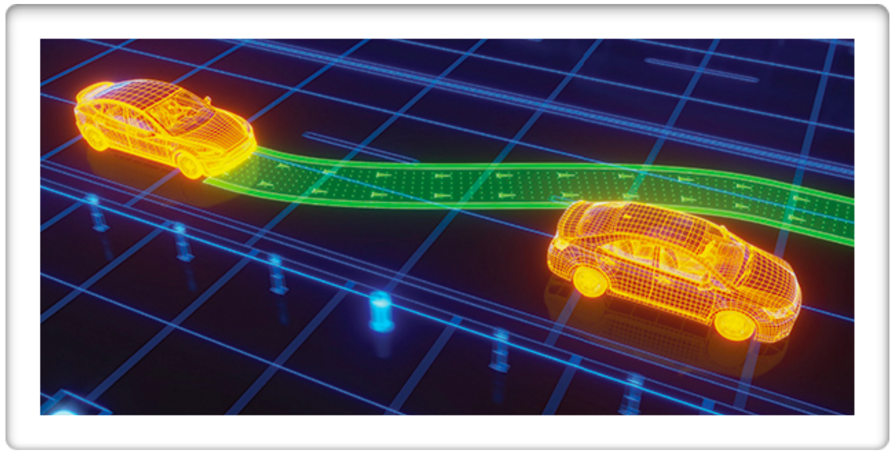
Hacker News — https://news.ycombinator.com

Hacker News — https://news.ycombinator.com/news

hacker news

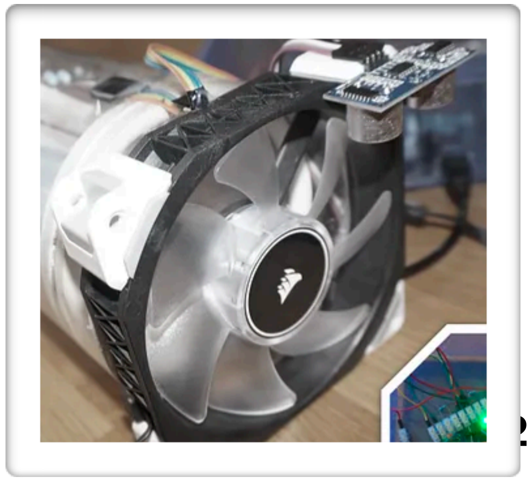
Firefox's URL bar suggestion

IoT



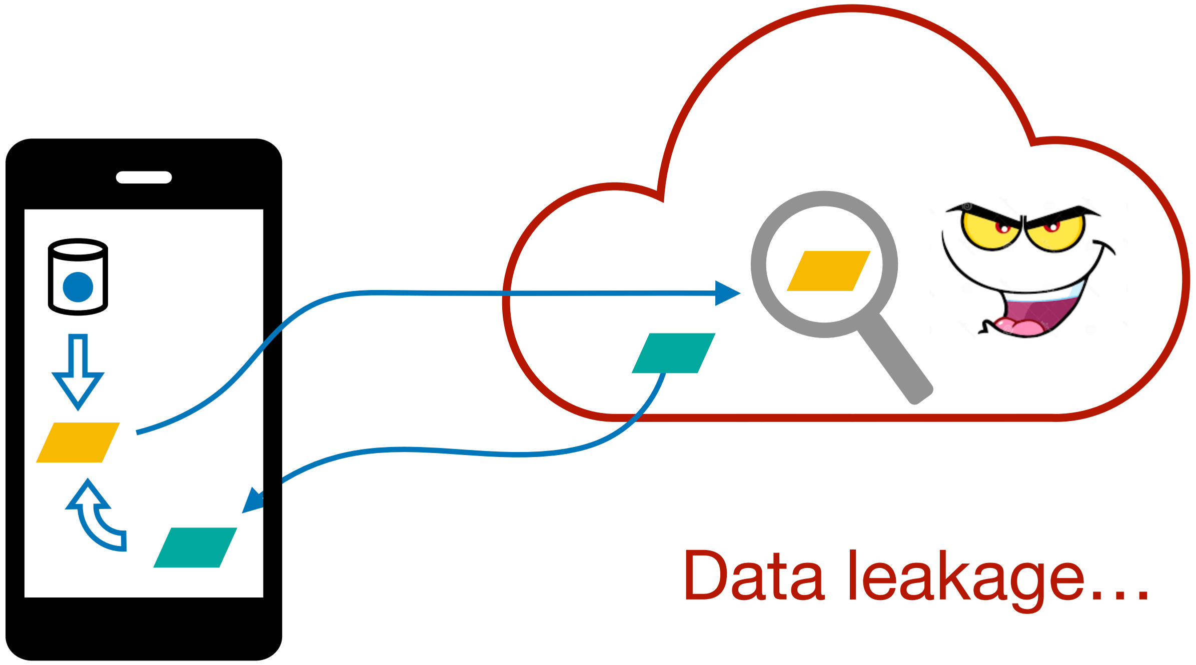
Volvo's trajectory prediction

Cisco's 3D printing



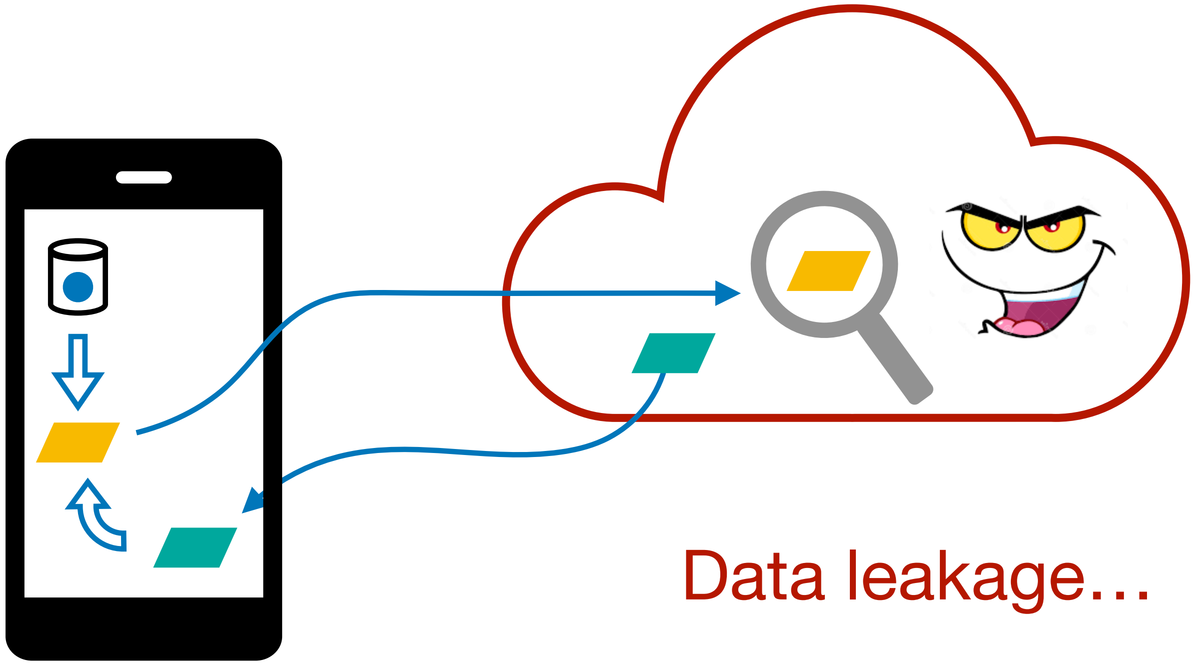
Leveno's clogging detection

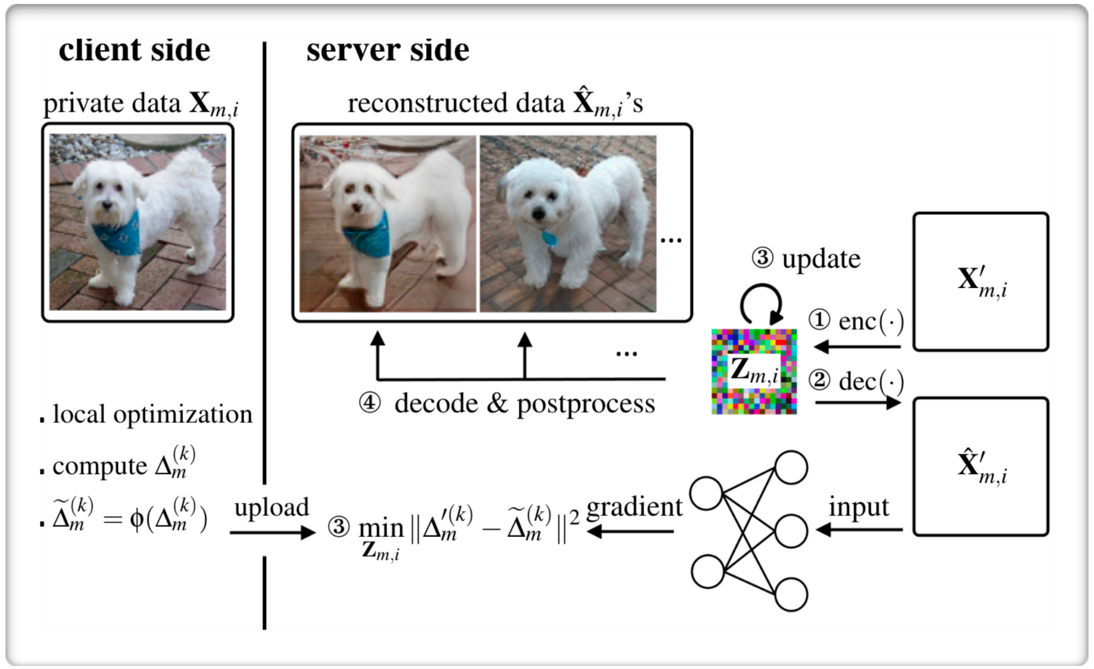
Challenge: identify and address the fundamental privacy and efficiency issues in cross-device FL



Data leakage…

Challenge: identify and address the fundamental privacy and efficiency issues in cross-device FL

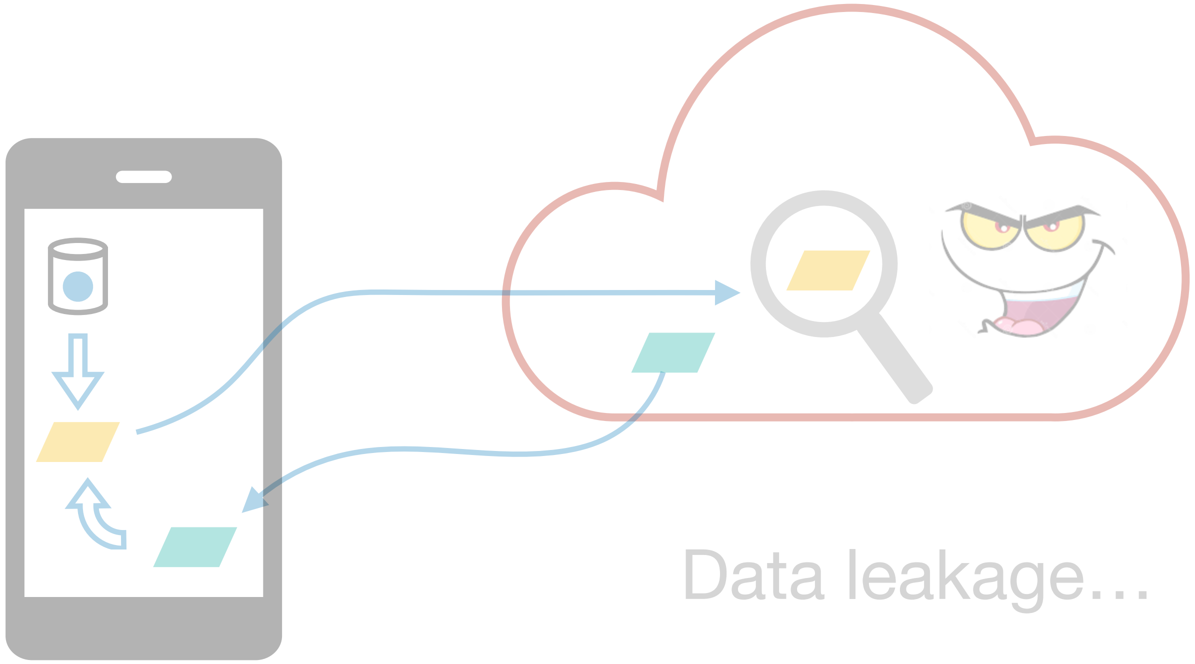



client side
server side
private data $\mathbf{X}_{m,i}$
reconstructed data $\hat{\mathbf{X}}_{m,i}$'s
...
③ update
$\mathbf{X}'_{m,i}$
① enc(·)
$\mathbf{Z}_{m,i}$
② dec(·)
④ decode & postprocess
. local optimization
. compute $\Delta_m^{(k)}$
. $\widetilde{\Delta}_m^{(k)} = \phi(\Delta_m^{(k)})$
upload
③ $\min_{\mathbf{Z}_{m,i}} \|\Delta_m'^{(k)} - \widetilde{\Delta}_m^{(k)}\|^2$
gradient
input
$\hat{\mathbf{X}}'_{m,i}$

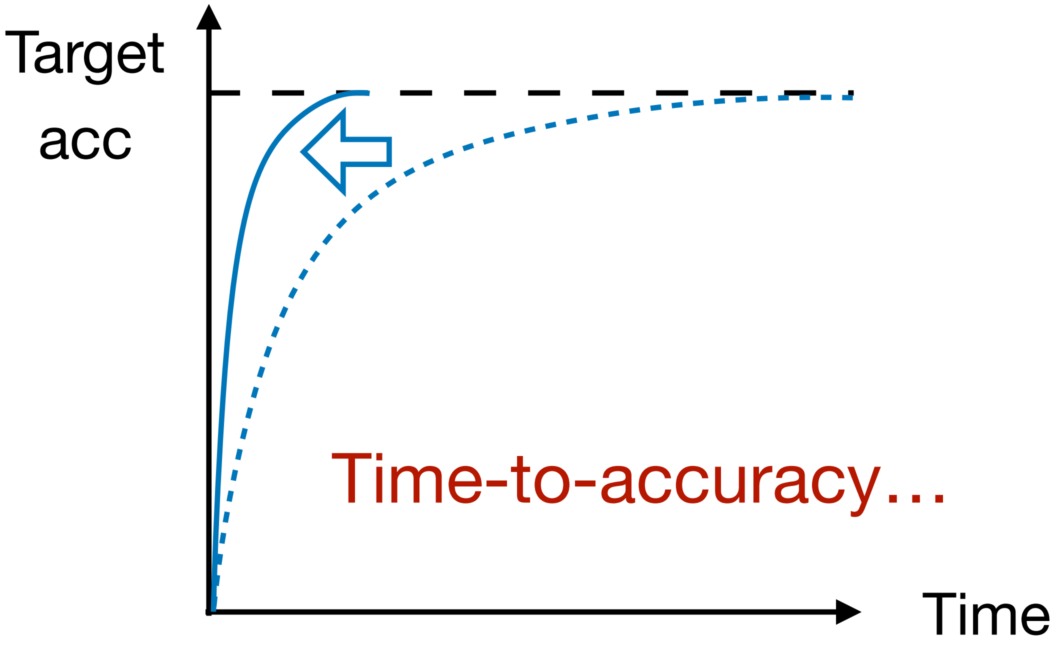
e.g., data reconstruction[1] (Security '23)

[1] Gradient Obfuscation Gives a False Sense of Security in Federated Learning

Challenge: identify and address the fundamental privacy and efficiency issues in cross-device FL

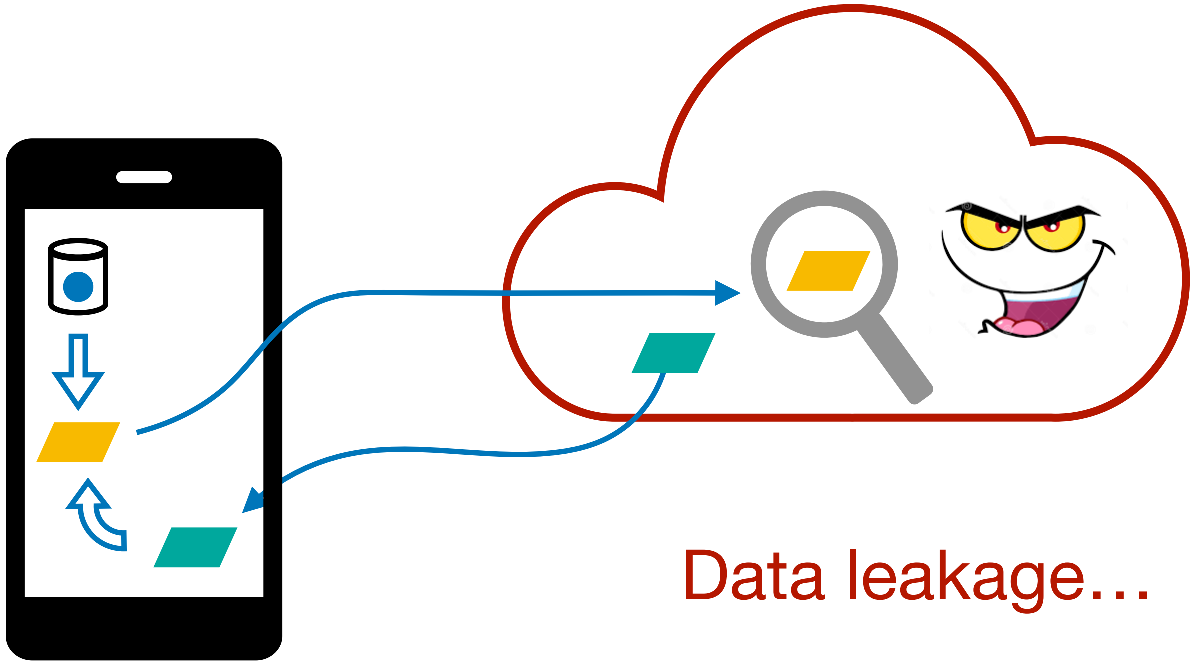


Data leakage…

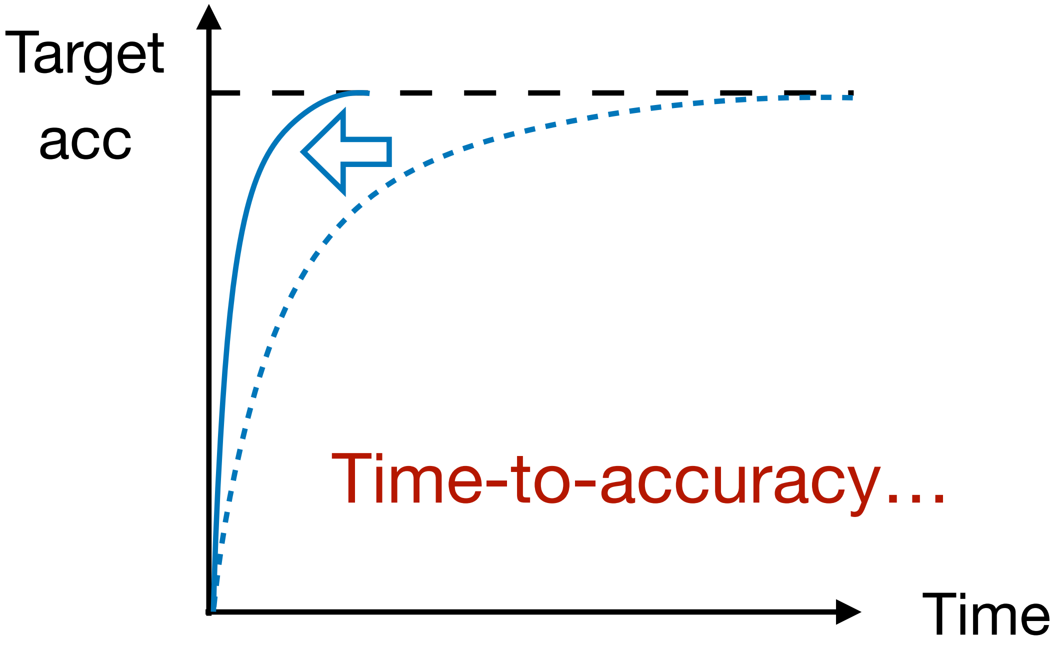


Time-to-accuracy…

My Work: build private and efficient cross-device FL



Data leakage…

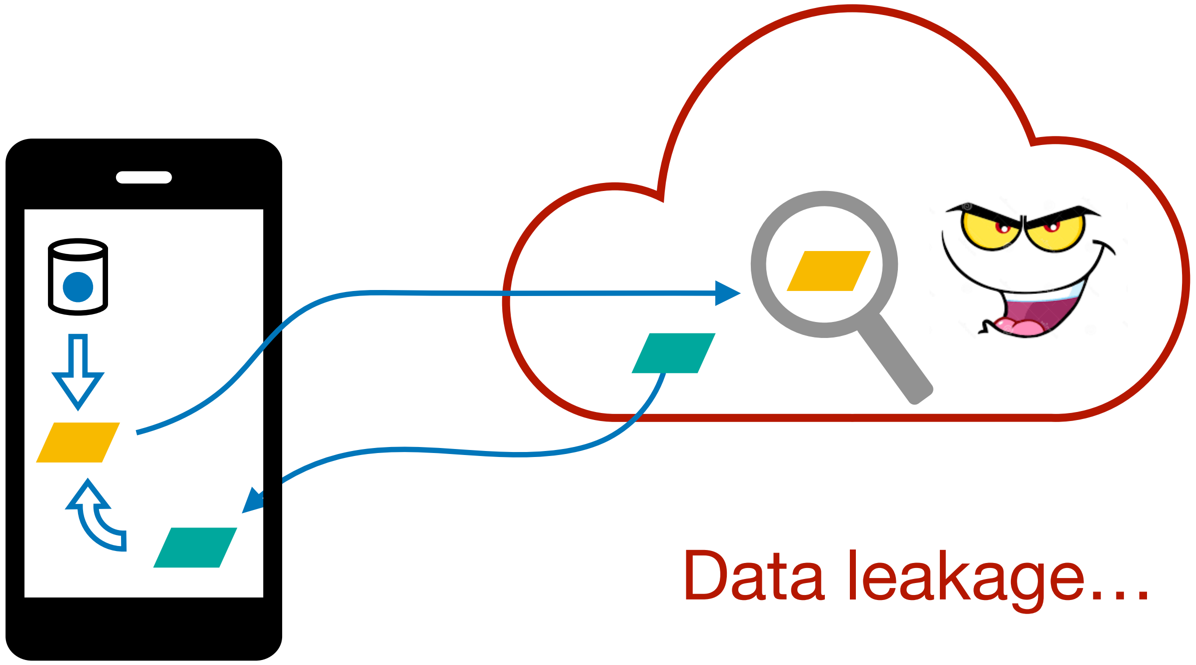


Time-to-accuracy…

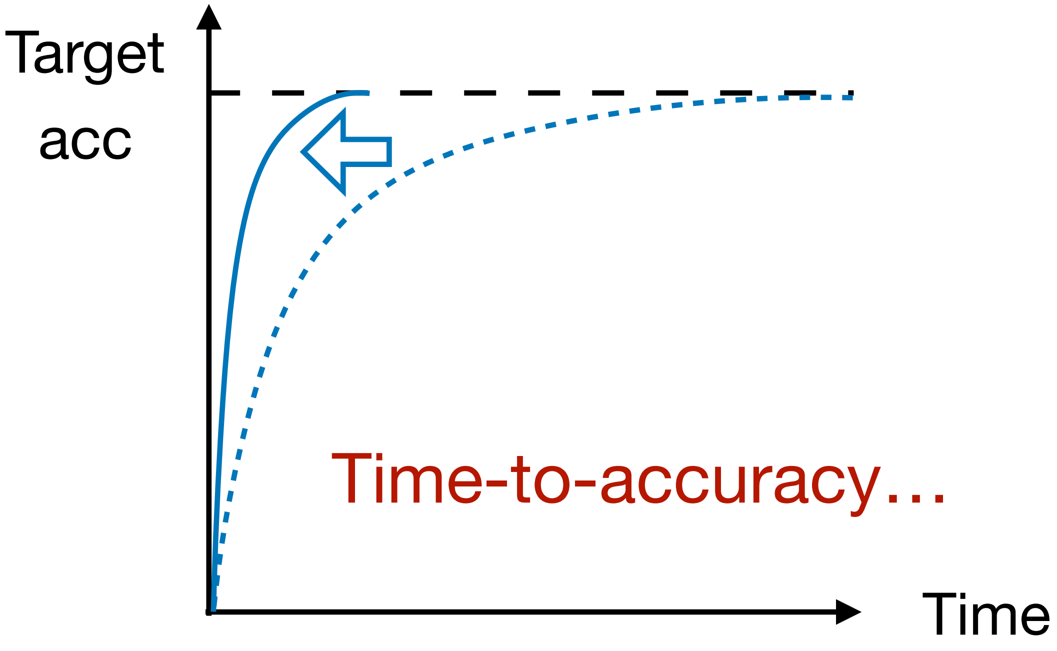
Weak privacy attackers

Efficient asynchronous training (SoCC ’22)

My Work: build private and efficient cross-device FL



Data leakage…



Time-to-accuracy…

Weak privacy attackers

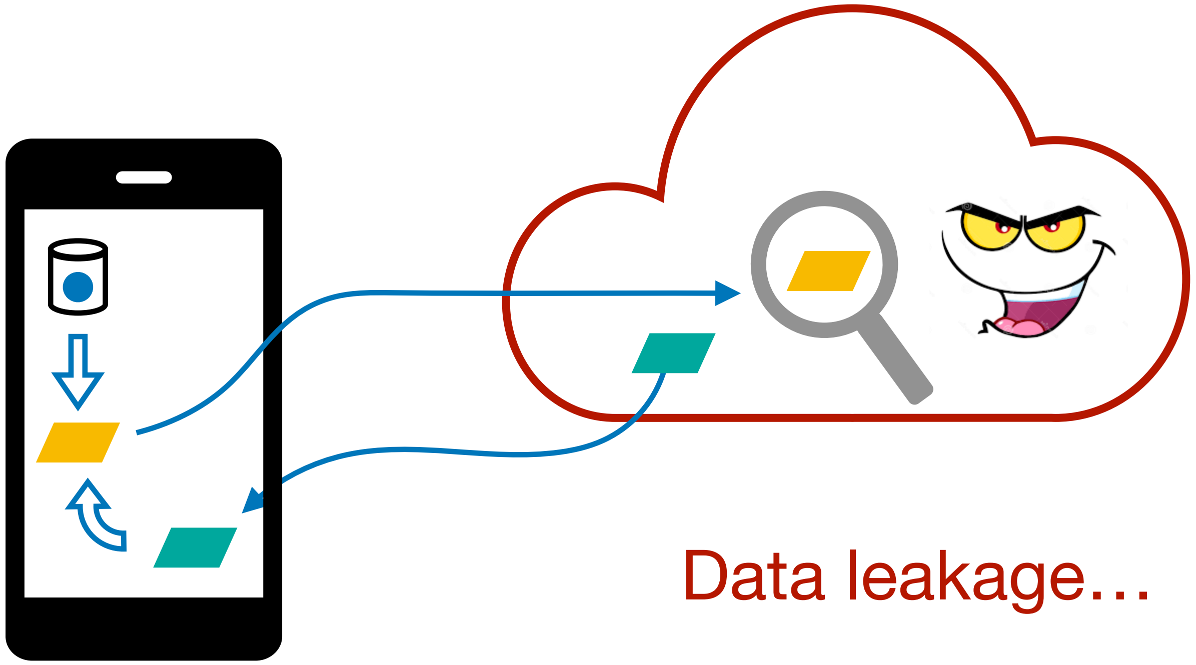
Efficient asynchronous training (SoCC ’22)

Dropout-resilient & pipeline-accelerated distributed differential privacy (EuroSys ’24)

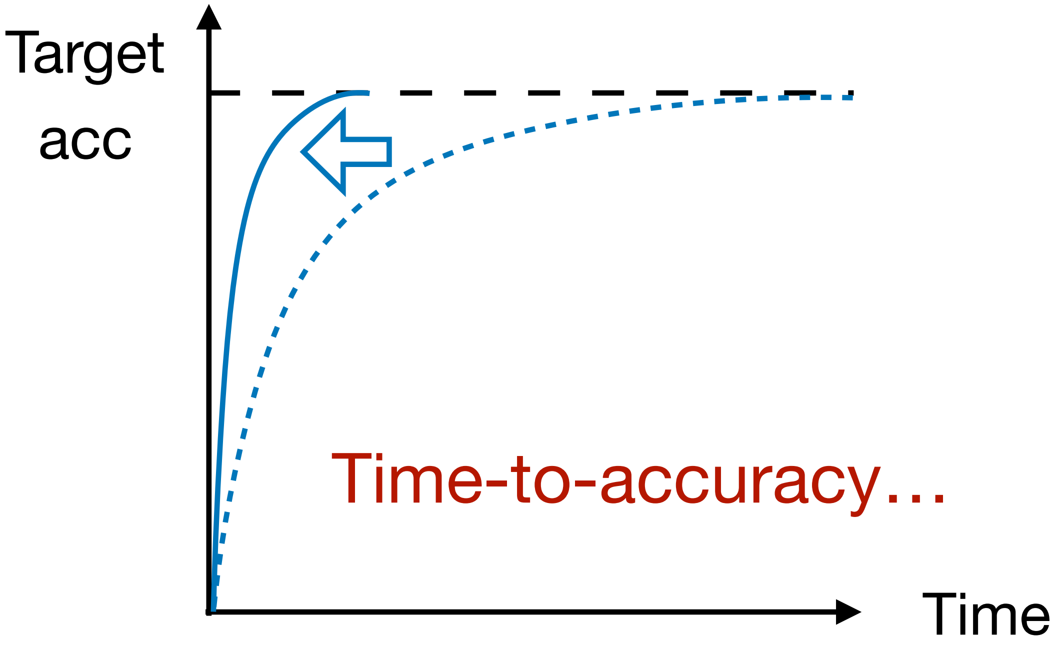
Secure participant selection (Security ’24)

Strong privacy attackers

My Work: build private and efficient cross-device FL



Data leakage…



Time-to-accuracy…

Weak privacy attackers

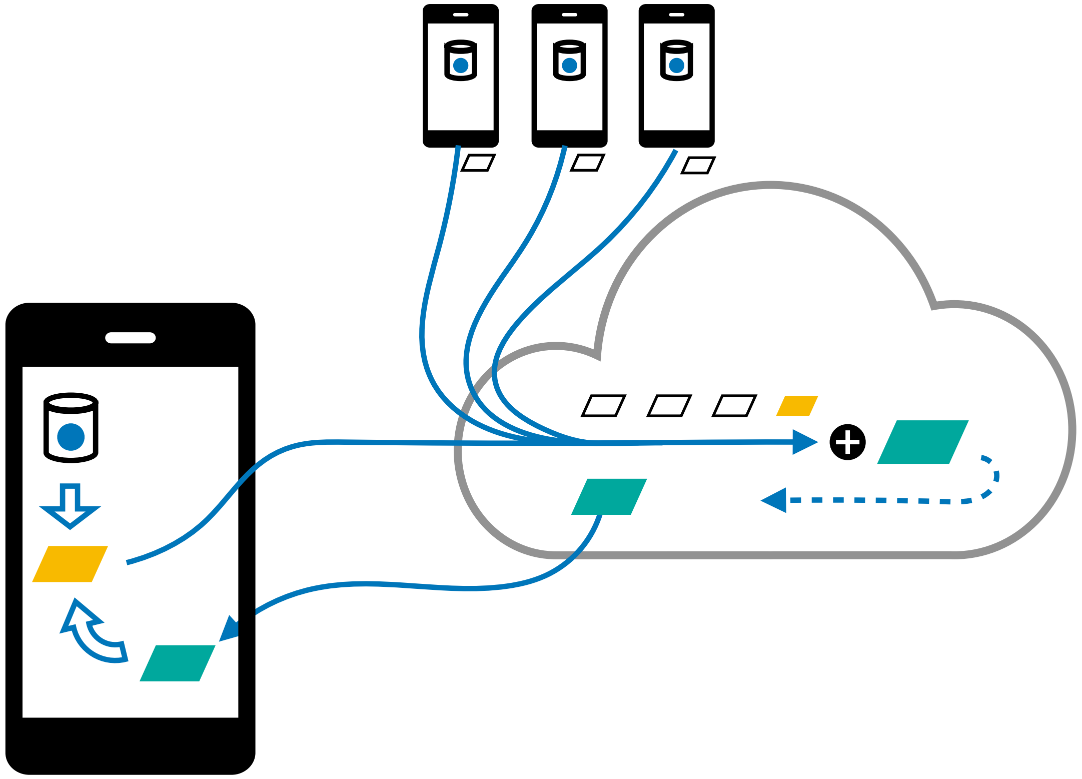
Efficient asynchronous training (SoCC ’22)

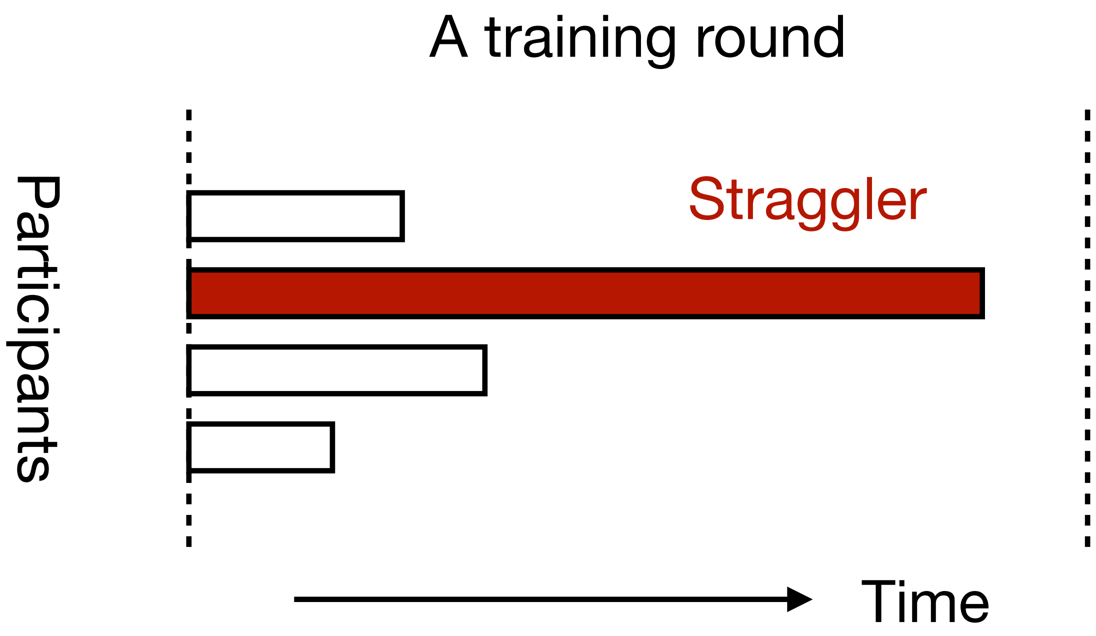
Dropout-resilient & pipeline-accelerated distributed differential privacy (EuroSys ’24)

Secure participant selection (Security ’24)

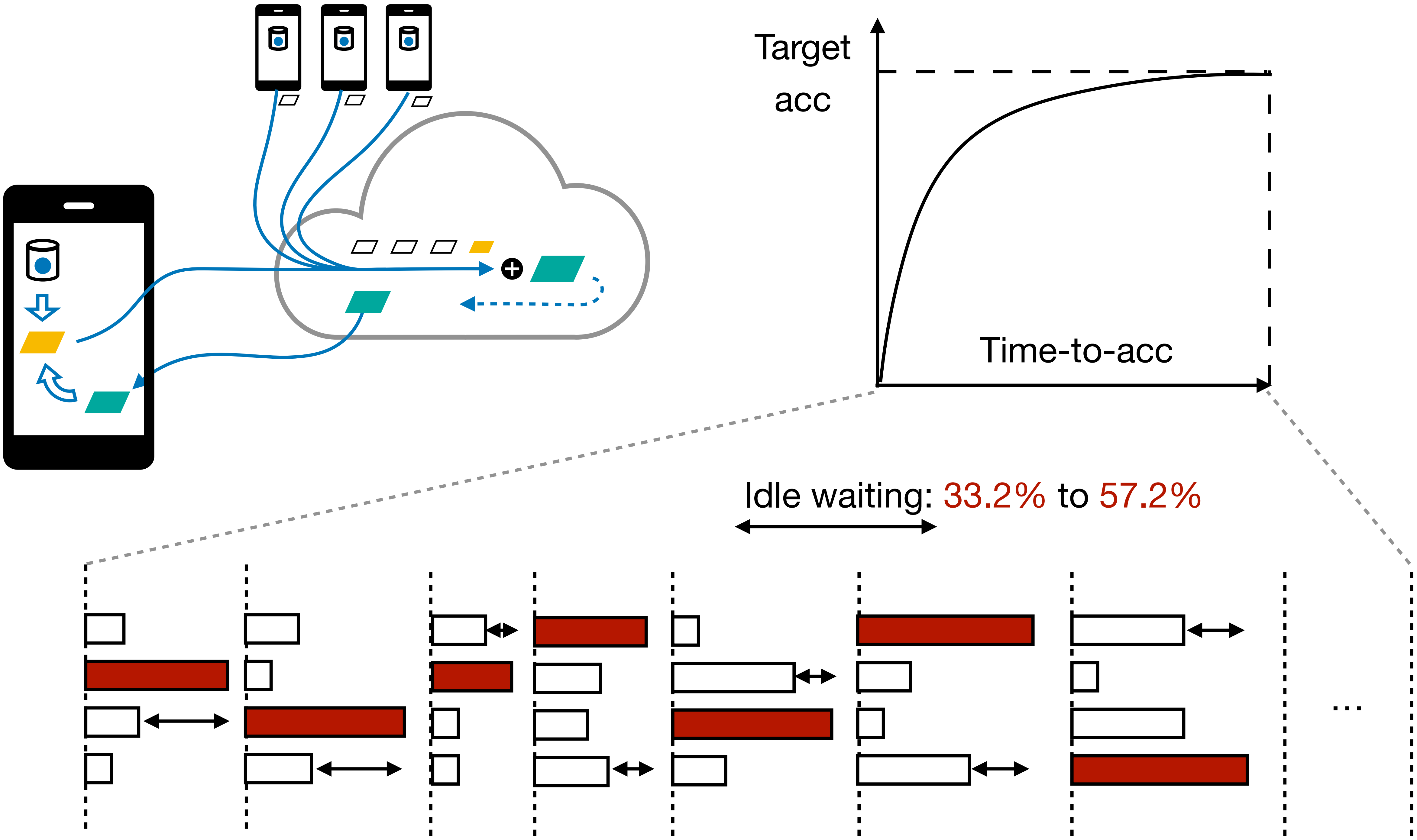
Strong privacy attackers

Stragglers are an efficiency bottleneck in sync FL





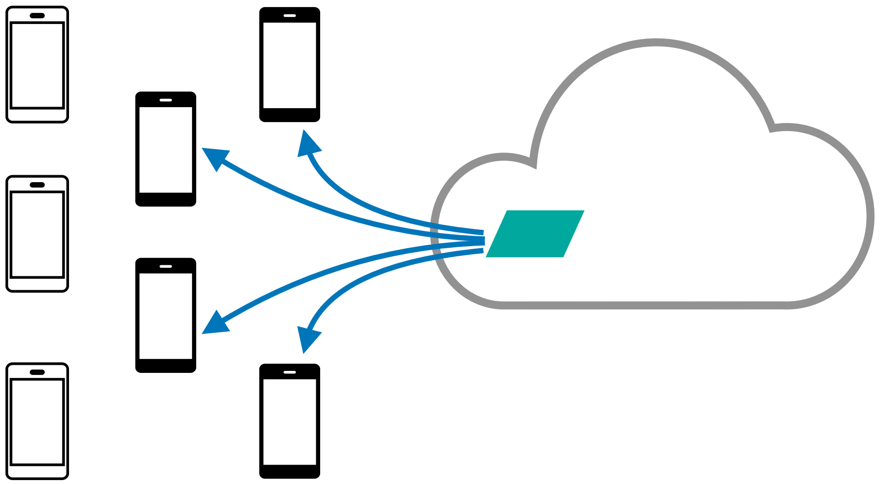
Stragglers are an efficiency bottleneck in sync FL

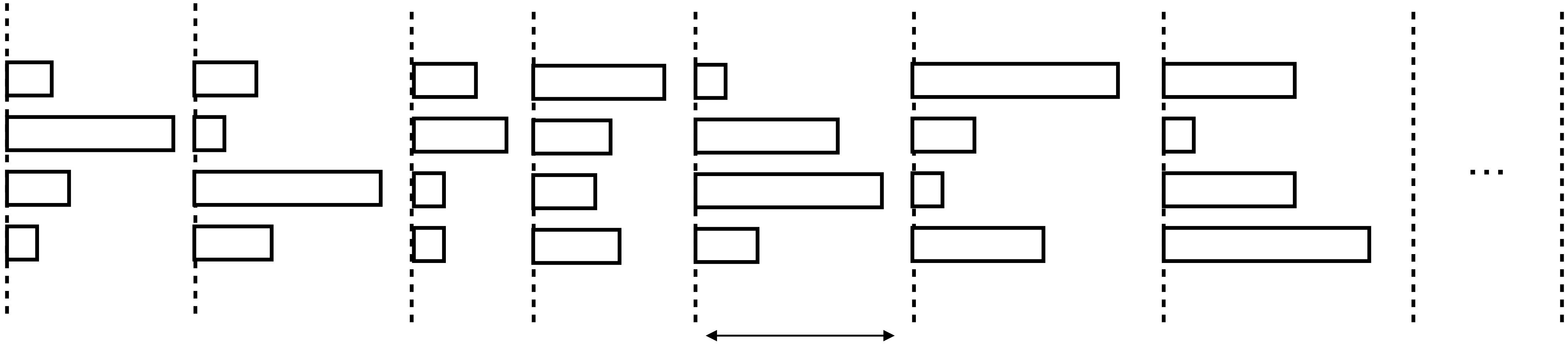


Idle waiting: 33.2% to 57.2%

Participant selection as a fix?

Prioritize clients with high speed

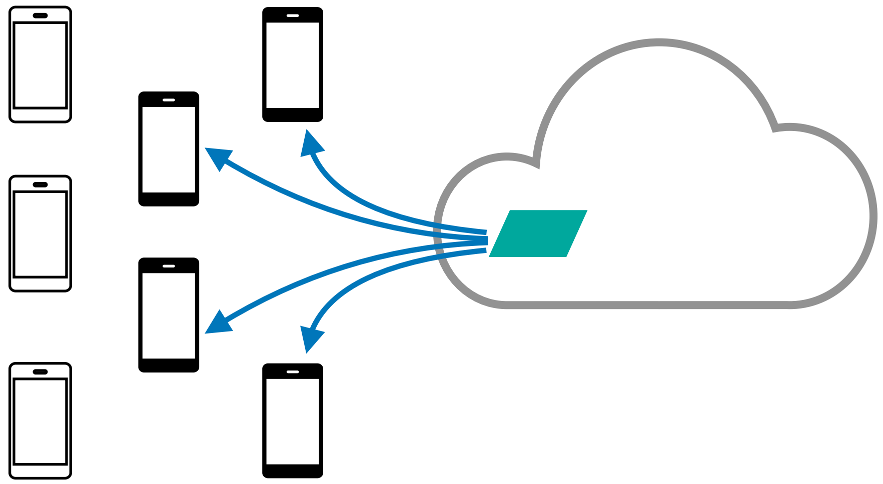




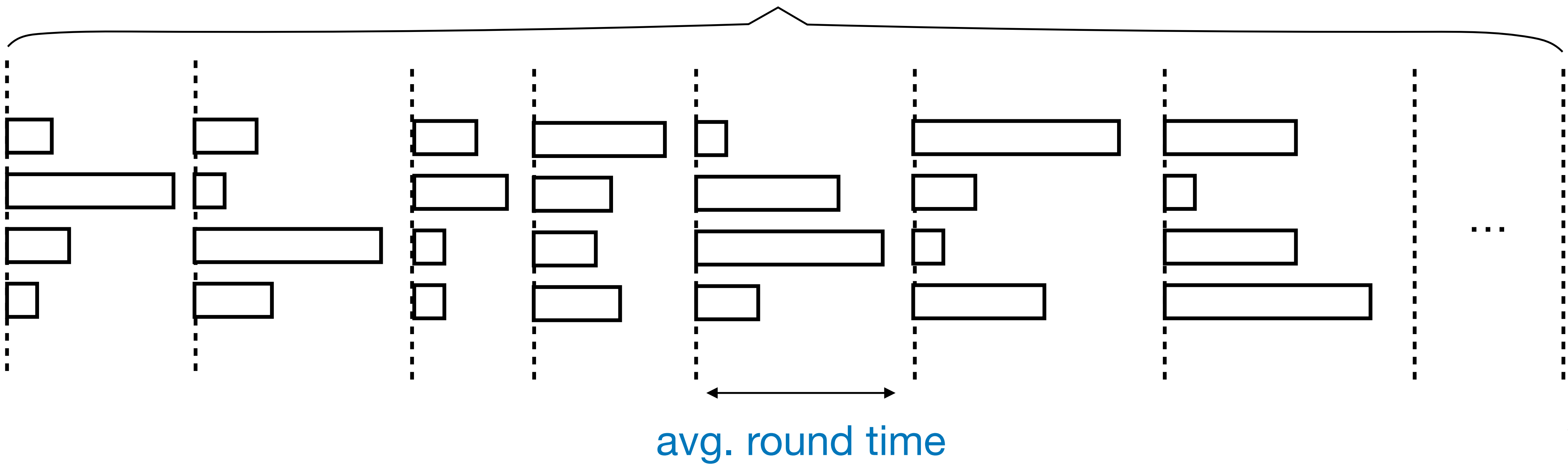
avg. round time ↓

Participant selection as a fix?

Prioritize clients with high speed and data quality



time-to-accuracy = [avg. round time] × [# rounds]



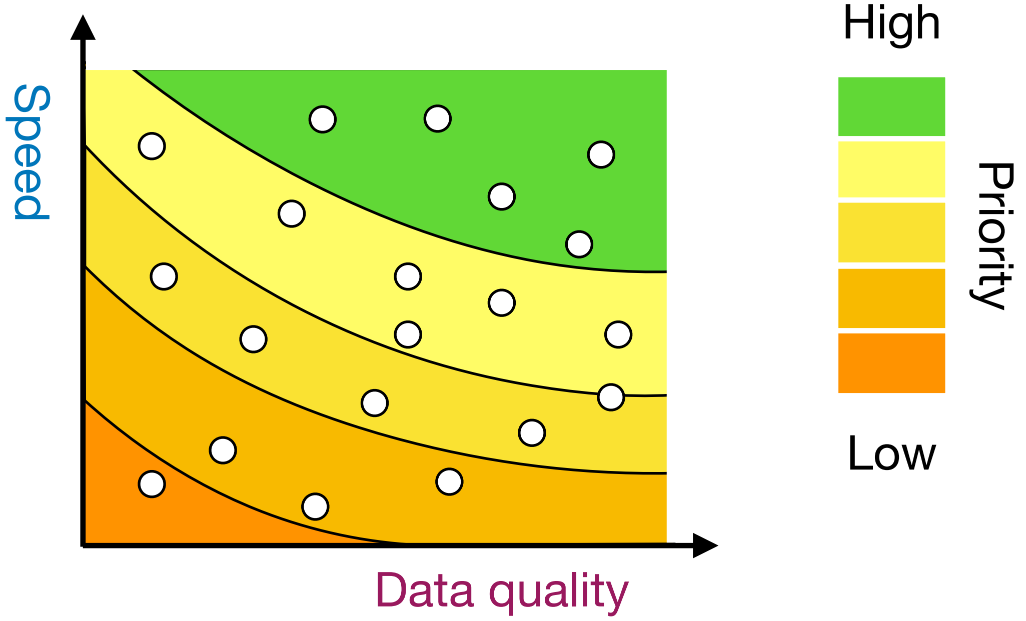
Participant selection as a fix?

Prioritize clients with high speed and data quality

State-of-the-art: **Oort**[1] (OSDI ’21)

- Clients with higher score are selected more
- Definition of score U_i for client i:

$$U_i = \underbrace{\left(\frac{T}{t_i}\right)^{\mathbf{1}(T<t_i)\times\alpha}}_{\text{speed}} \times \underbrace{|B_i|\sqrt{\frac{1}{|B_i|}\sum_{k\in B_i} Loss(k)^2}}_{\text{data quality}}$$



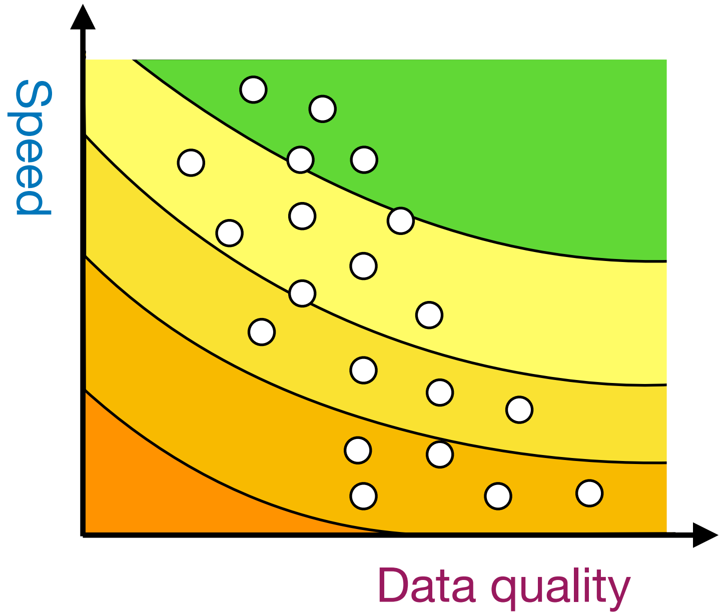
[1] Oort: Efficient federated learning via guided participant selection

Participant selection as a fix?

Prioritize clients with high speed and data quality

State-of-the-art: **Oort** (OSDI ’21)

Inefficient in achieving the best tradeoff in practice where speed $\propto \frac{1}{\text{data quality}}$

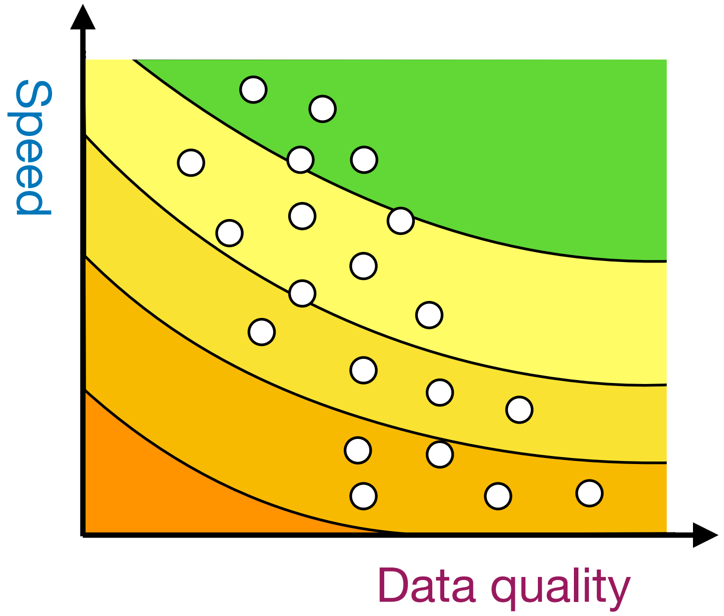


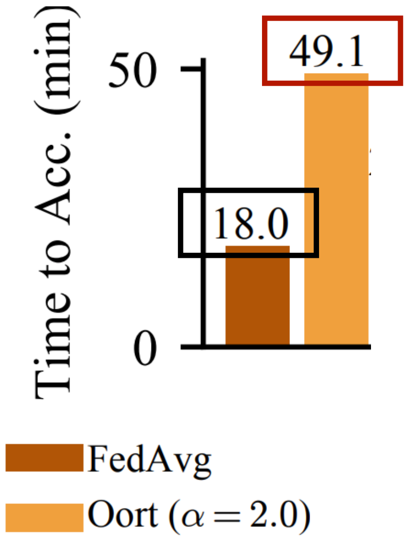
Participant selection as a fix?

Prioritize clients with high speed and data quality

State-of-the-art: **Oort** (OSDI ’21)

Inefficient in achieving the best tradeoff in practice where speed $\propto \frac{1}{\text{data quality}}$





2.7× slower than random selection!

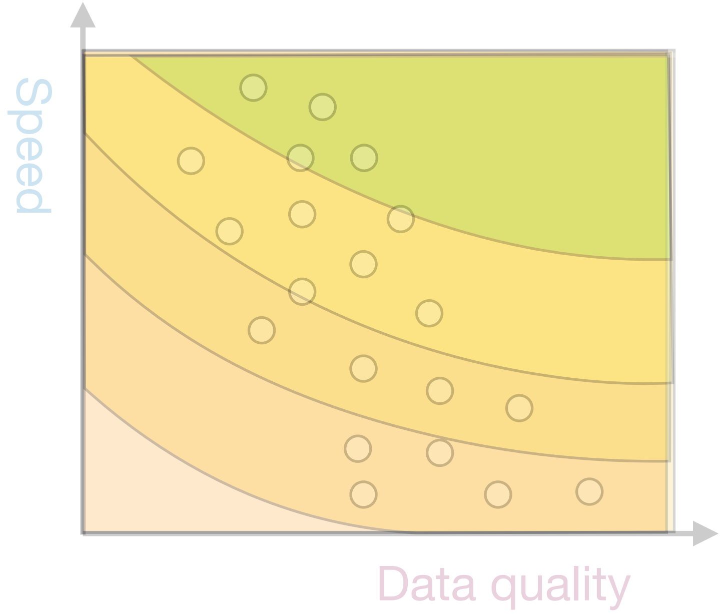
an FL testbed

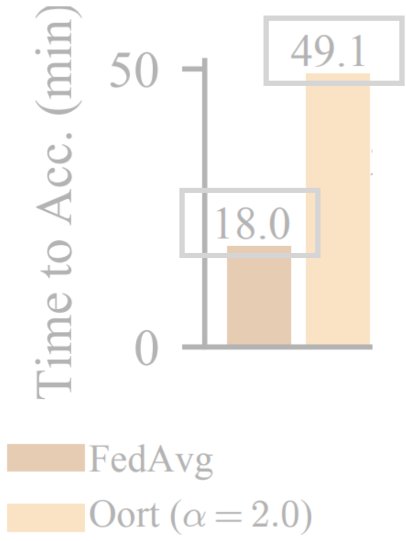
Participant selection as a fix?

Prioritize clients with high speed and data quality

State-of-the-art: **Oort** (OSDI ’21)

Inefficient in achieving the best tradeoff in practice where speed $\propto \frac{1}{\text{data quality}}$





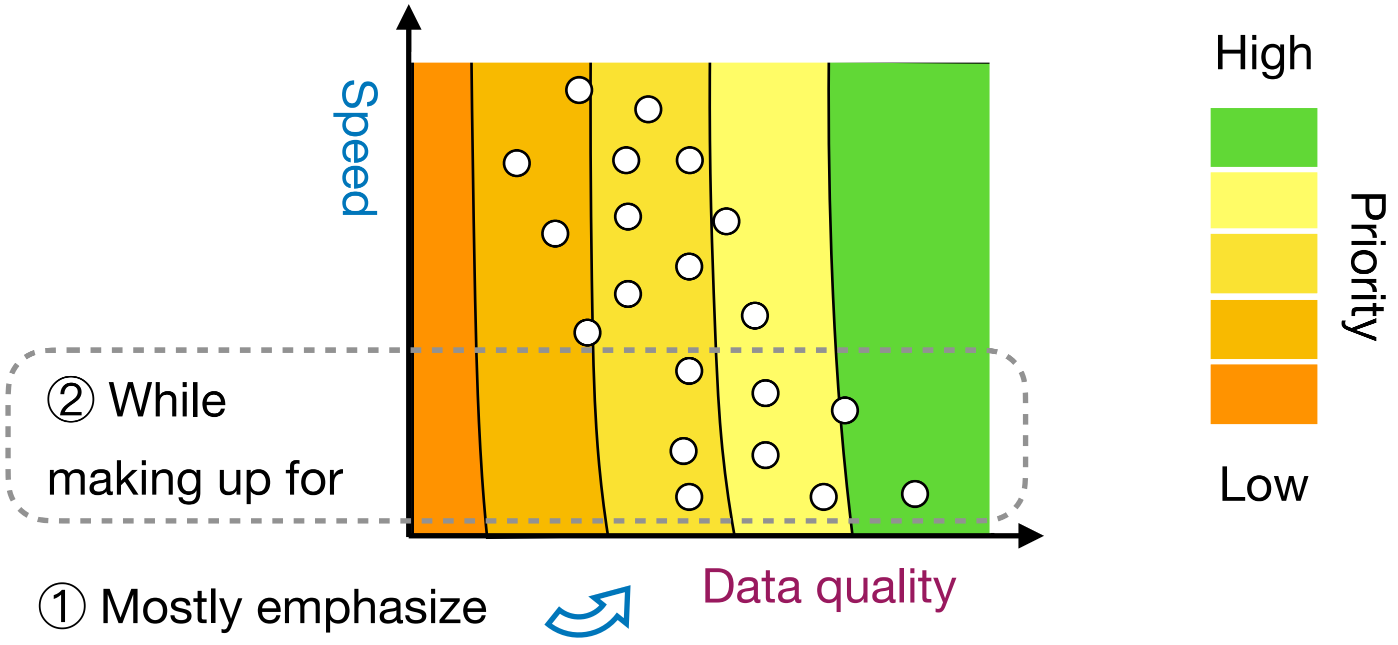
2.7× slower than random selection!

an FL testbed

Fundamental challenge in sync FL: unpleasant coupling demands for speed and data quality

To sidestep this challenge

Can we decouple them?

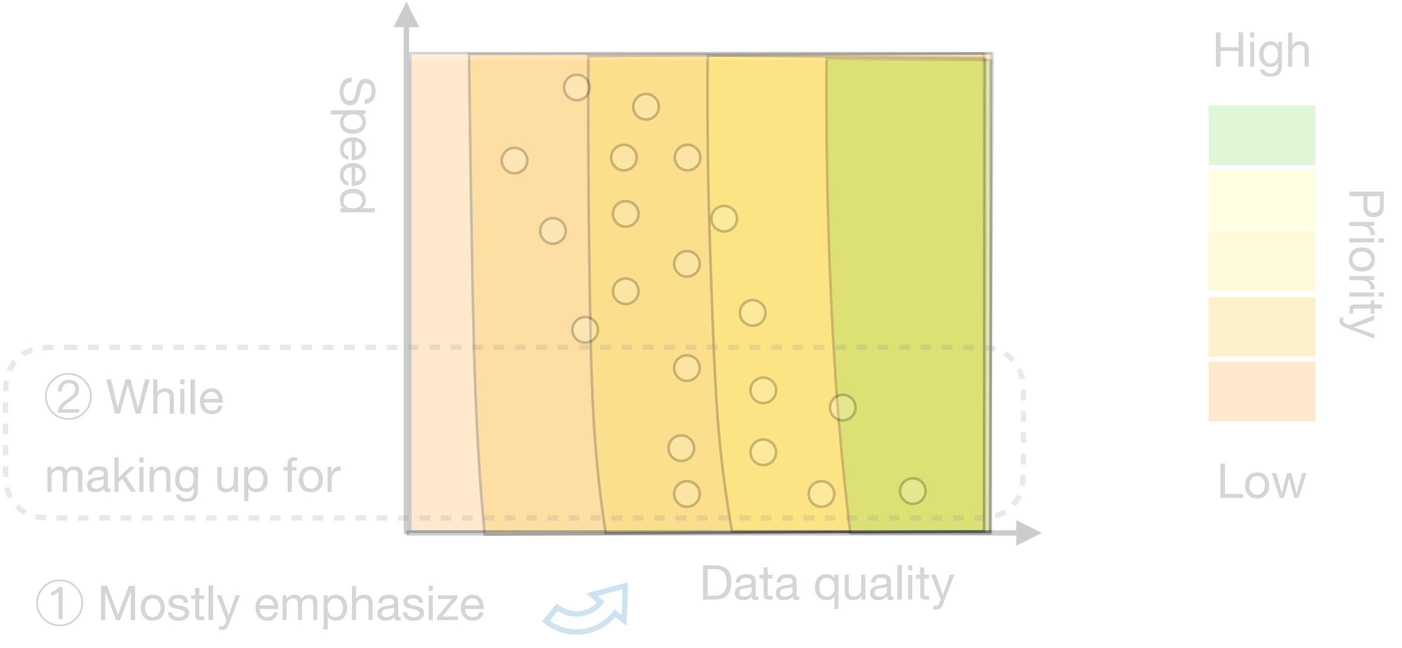


② While making up for

① Mostly emphasize

To sidestep this challenge

Can we decouple them?

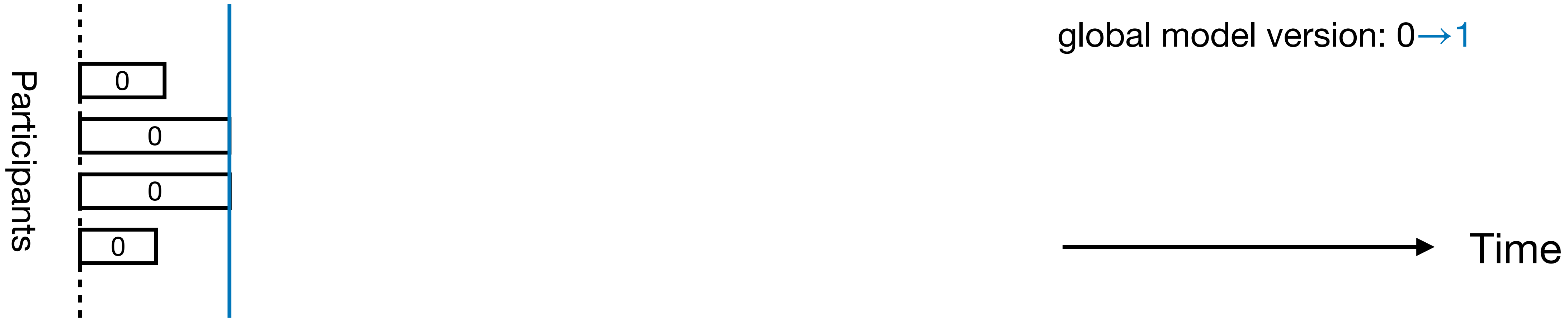


Sure! If the training is **asynchronous**

To sidestep this challenge

Asynchronous Training

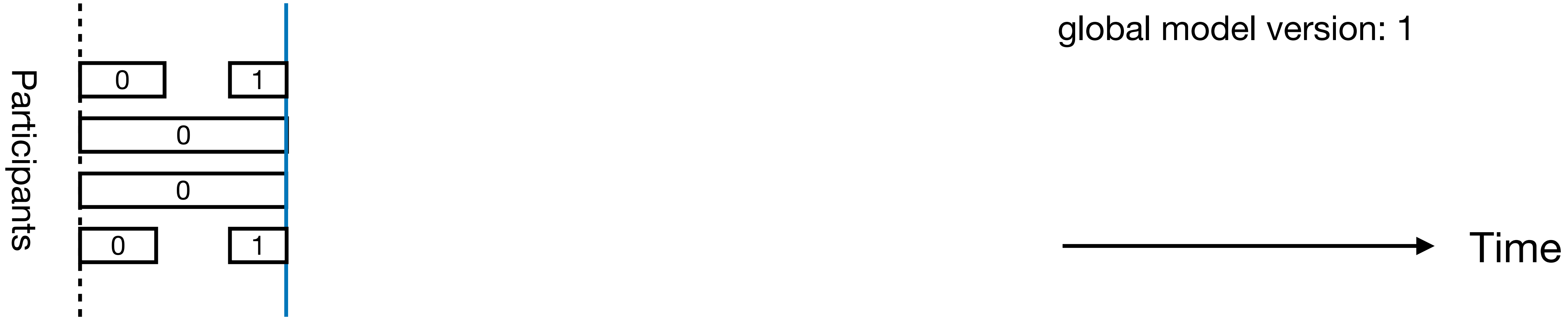
- Select some clients with **best data** and send them the latest model
- Early aggregate local updates **without waiting** for some running participants



To sidestep this challenge

Asynchronous Training

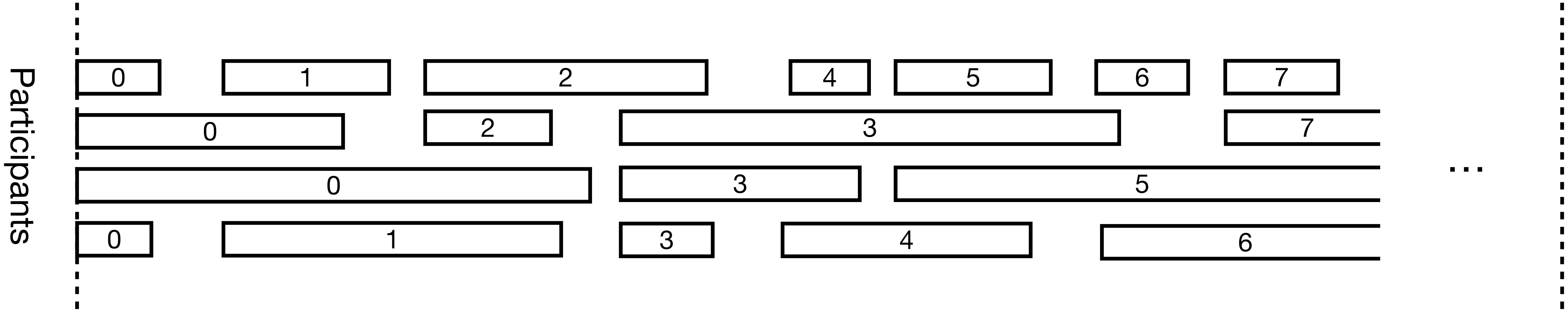
- Select some clients with **best data** and send them the latest model
- Early aggregate local updates **without waiting** for some running participants



To sidestep this challenge

Asynchronous Training

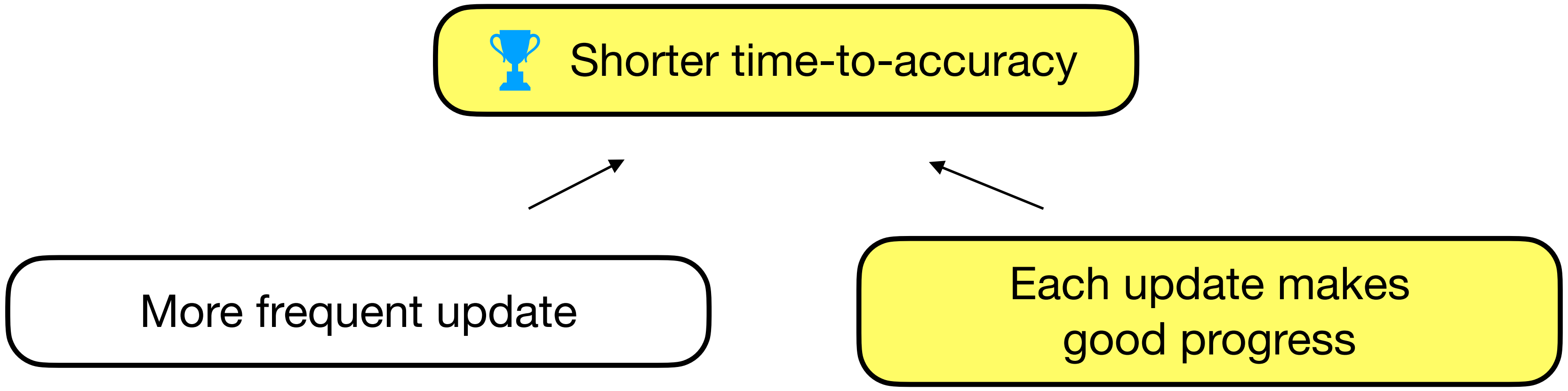
- Select some clients with **best data** and send them the latest model
- Early aggregate local updates **without waiting** for some running participants



More frequent update

To sidestep this challenge

How to really benefit efficiency with async FL?



🏆 Shorter time-to-accuracy

More frequent update

Each update makes good progress

Done by Async FL

#TODO

To sidestep this challenge

How to really benefit efficiency with async FL?

🏆 Shorter time-to-accuracy

More frequent update

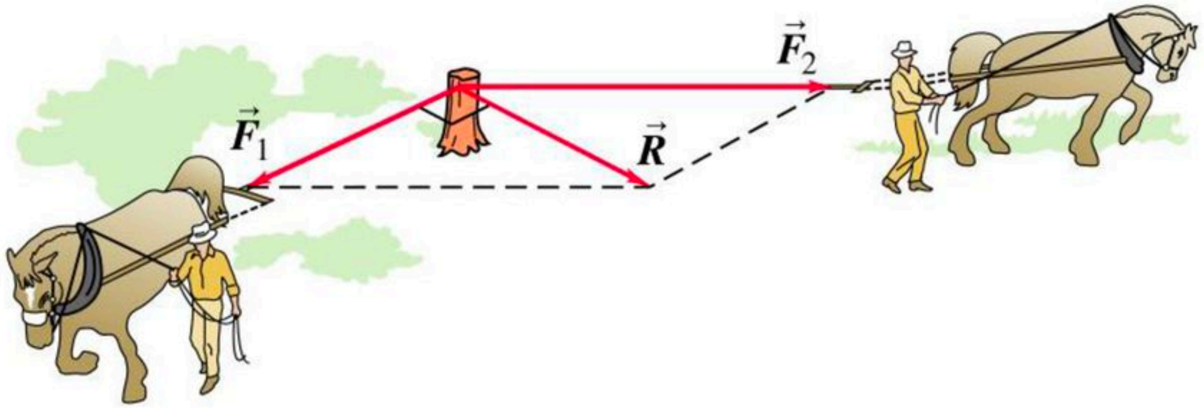
Each update makes good progress

Done by Async FL

The involved clients have good data

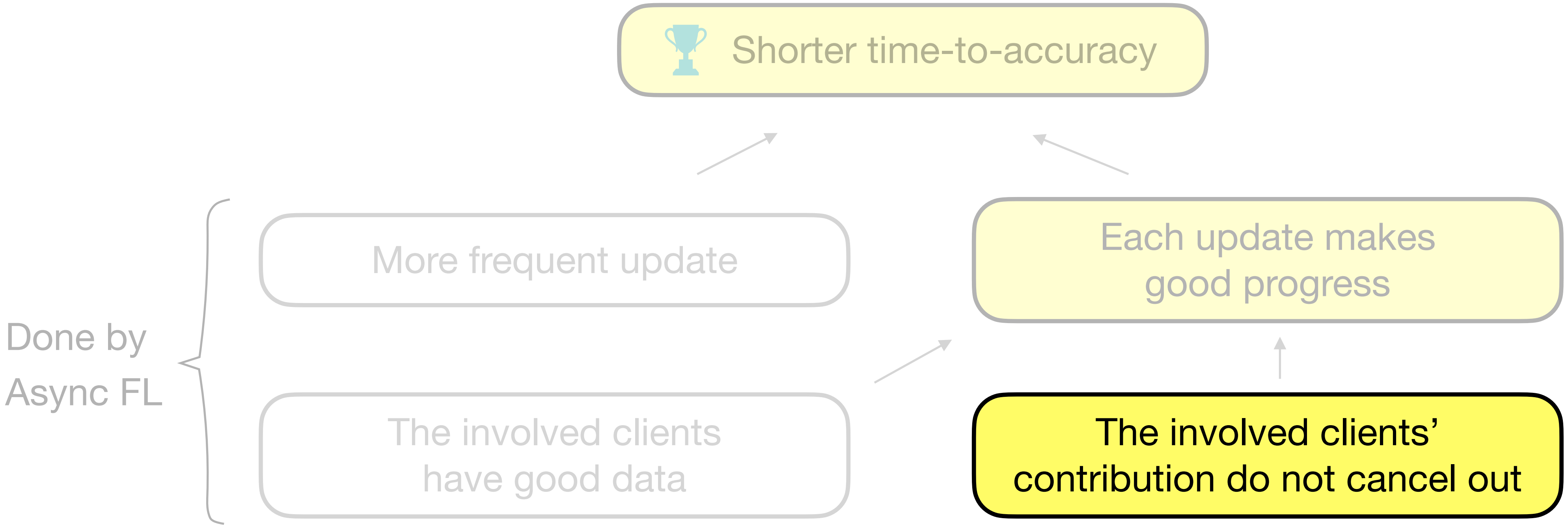
The involved clients' contribution do not cancel out

#TODO



To sidestep this challenge

How to really benefit efficiency with async FL?



Shorter time-to-accuracy

Done by Async FL

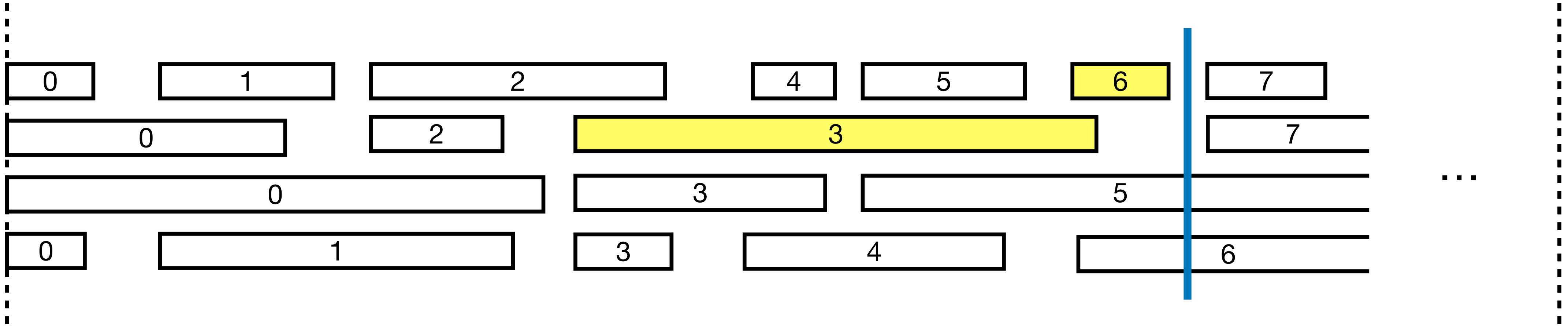
More frequent update

The involved clients have good data

Each update makes good progress

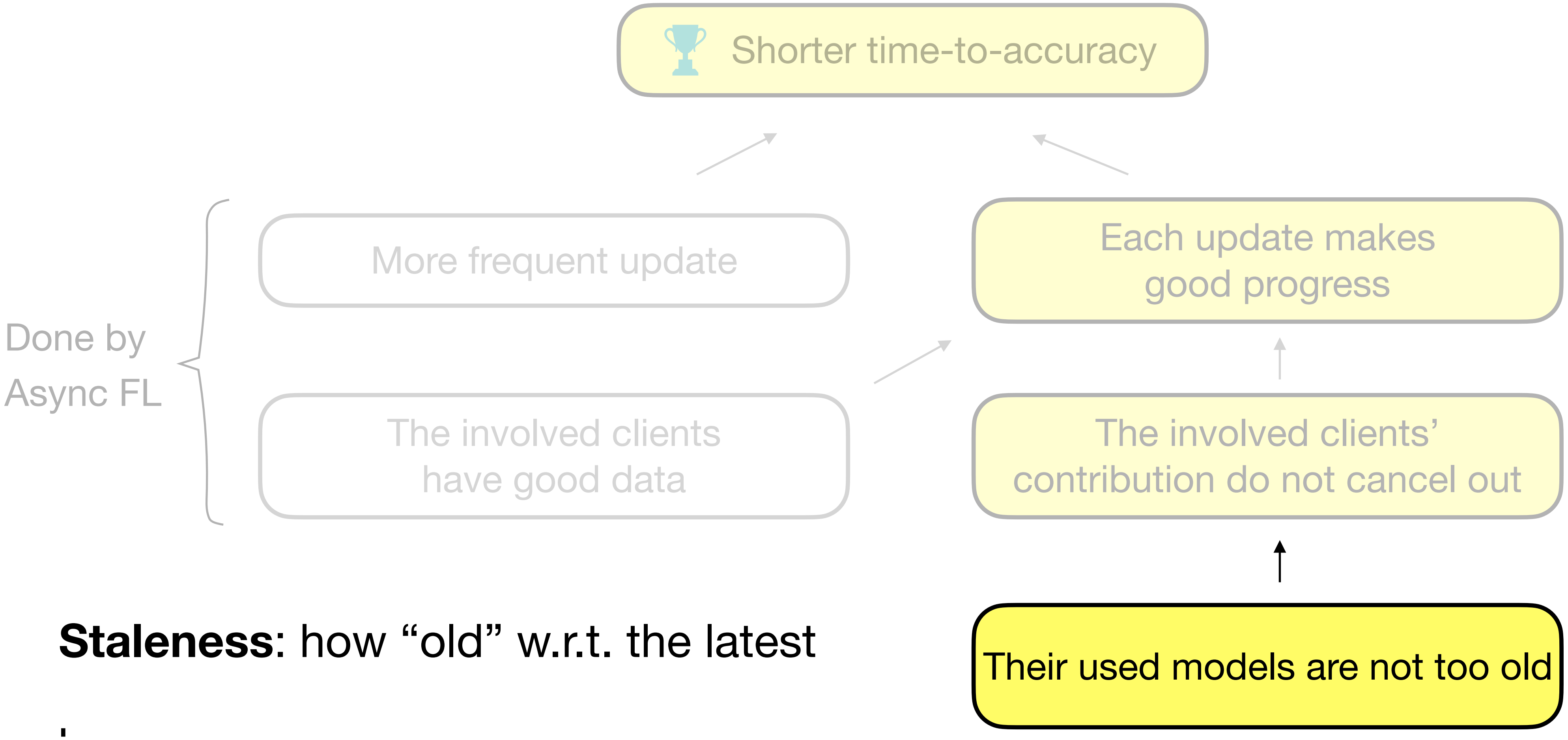
The involved clients' contribution do not cancel out

#TODO



To sidestep this challenge

How to really benefit efficiency with async FL?



🏆 Shorter time-to-accuracy

More frequent update

The involved clients have good data

Done by Async FL

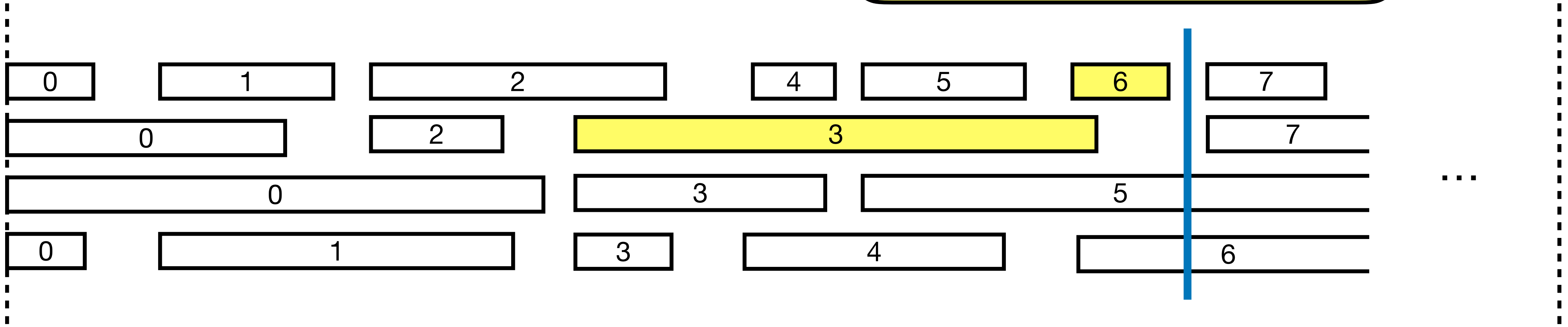
Each update makes good progress

The involved clients' contribution do not cancel out

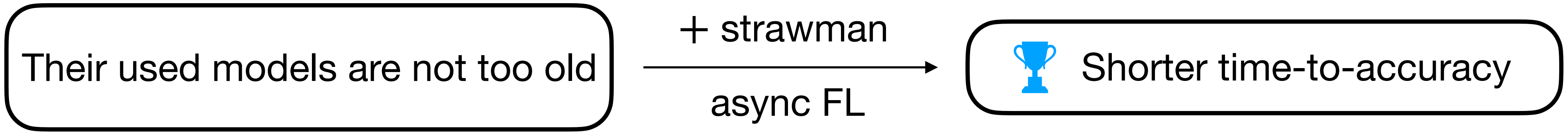
Their used models are not too old

Staleness: how "old" w.r.t. the latest

#TODO



Pisces: guided async FL with controlled staleness



Their used models are not too old

+ strawman

async FL

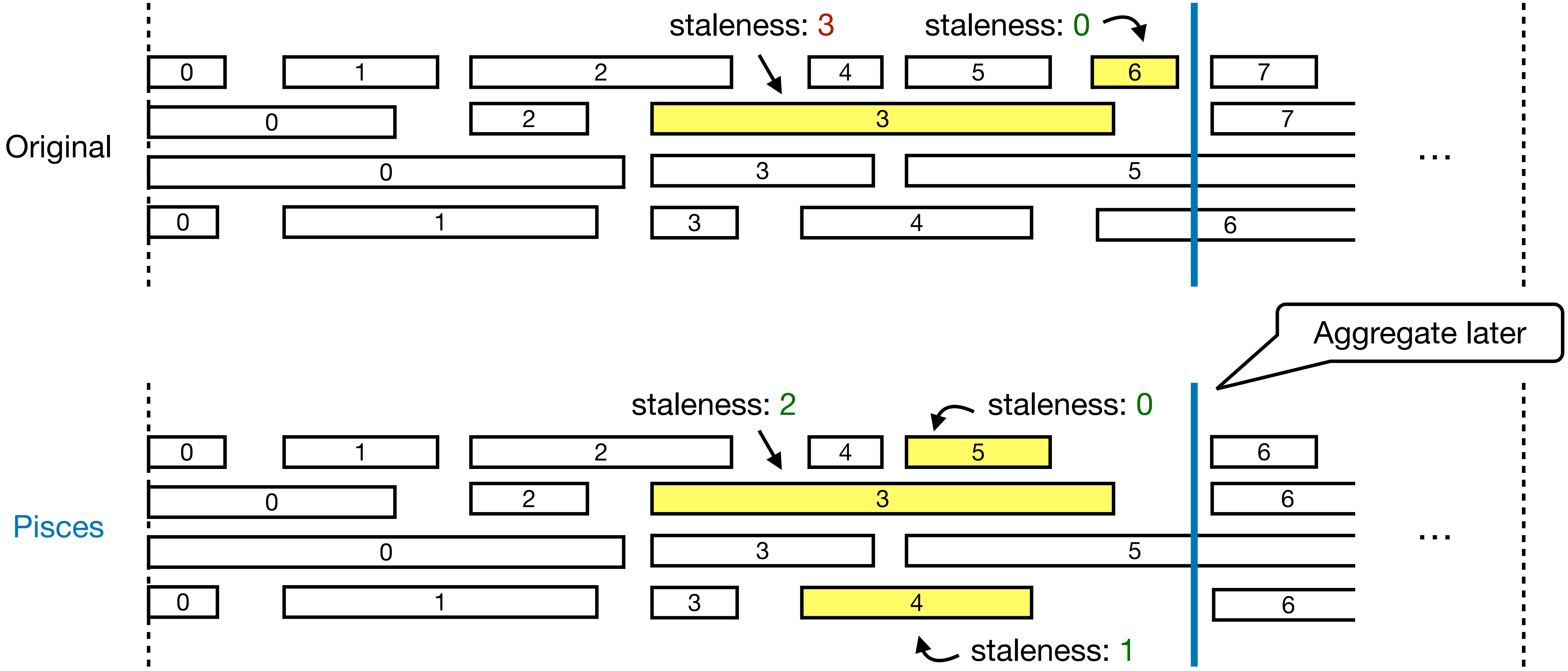
🏆 Shorter time-to-accuracy

① **Hard limit** on staleness

Pisces: guided async FL with controlled staleness

Their used models are not too old → + strawman async FL → 🏆 Shorter time-to-accuracy

① **Hard limit** on staleness via pace control at model aggregation



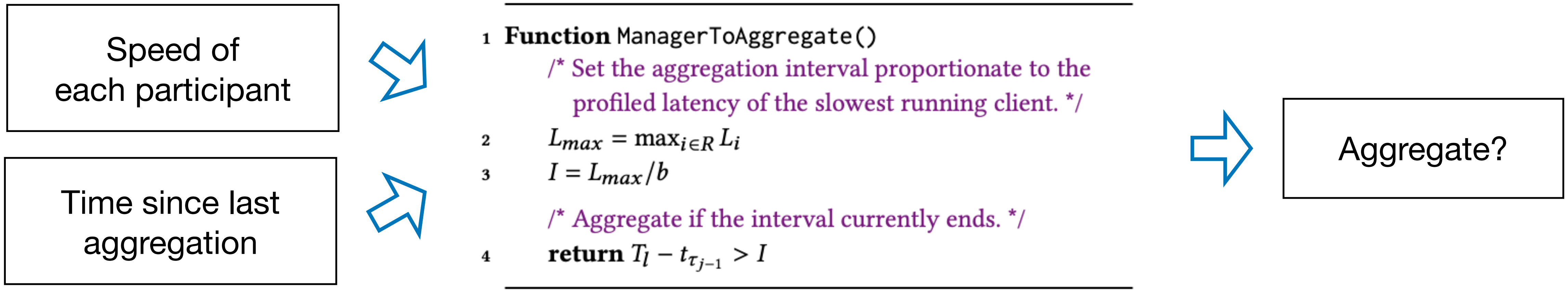
if the upper bound is 2

Pisces: guided async FL with controlled staleness

Their used models are not too old → + strawman async FL → 🏆 Shorter time-to-accuracy

① **Hard limit** on staleness via pace control at model aggregation

- Achieved by a neat yet provably effective algorithm



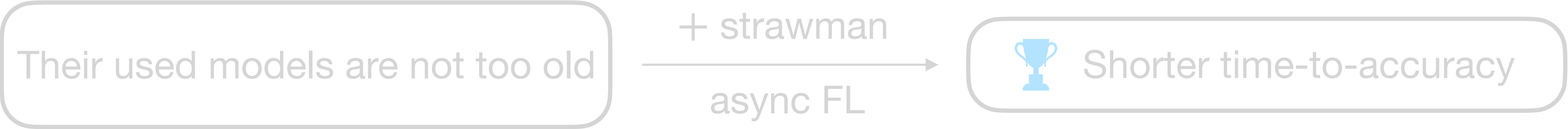
Speed of each participant

Time since last aggregation

```
Function ManagerToAggregate()
    /* Set the aggregation interval proportionate to the
       profiled latency of the slowest running client. */
    $L_{max} = \max_{i \in R} L_i$
    $I = L_{max}/b$
    /* Aggregate if the interval currently ends. */
    return $T_l - t_{\tau_{j-1}} > I$
```

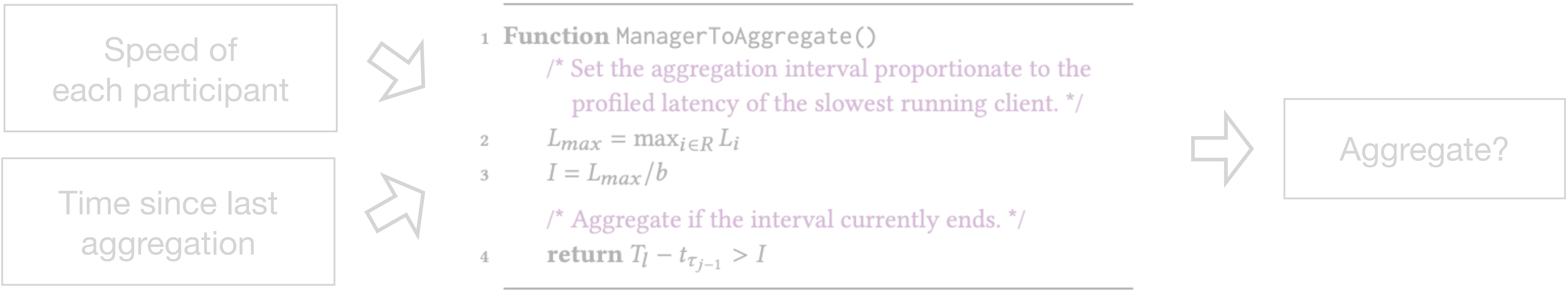
Aggregate?

Pisces: guided async FL with controlled staleness

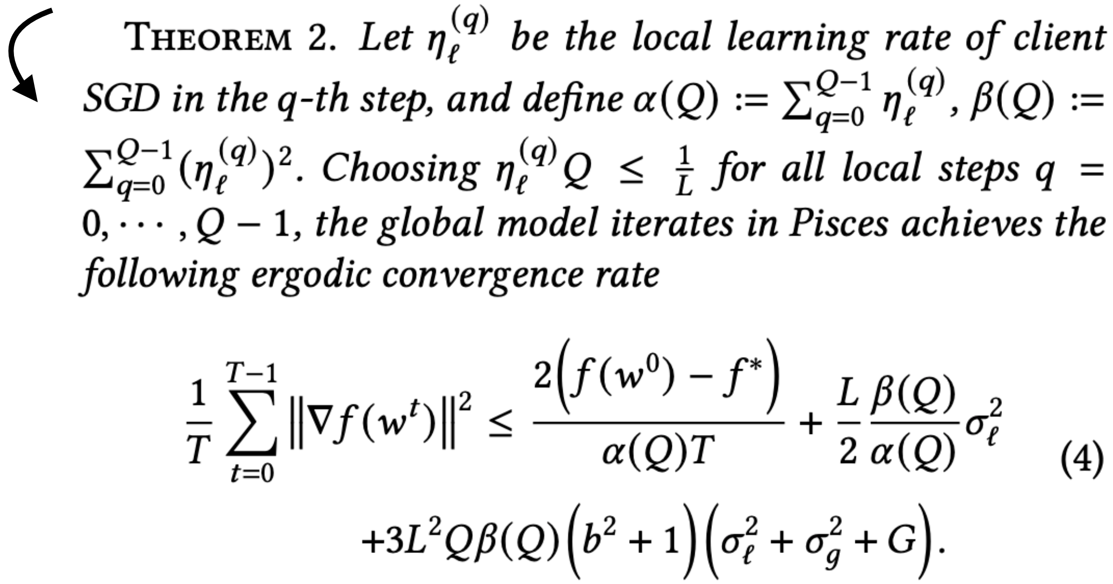


① **Hard limit** on staleness via pace control at model aggregation

- Achieved by a neat yet provably effective algorithm



```
Function ManagerToAggregate()
    /* Set the aggregation interval proportionate to the
       profiled latency of the slowest running client. */
    $L_{max} = \max_{i \in R} L_i$
    $I = L_{max}/b$
    /* Aggregate if the interval currently ends. */
    return $T_l - t_{\tau_{j-1}} > I$
```

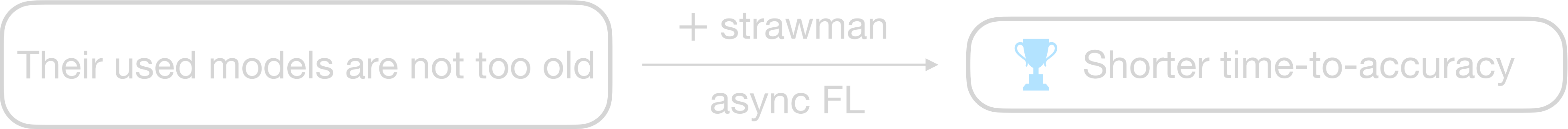


Theorem 2. *Let $\eta_\ell^{(q)}$ be the local learning rate of client SGD in the q-th step, and define $\alpha(Q) := \sum_{q=0}^{Q-1} \eta_\ell^{(q)}$, $\beta(Q) := \sum_{q=0}^{Q-1} (\eta_\ell^{(q)})^2$. Choosing $\eta_\ell^{(q)} Q \leq \frac{1}{L}$ for all local steps $q = 0, \cdots, Q-1$, the global model iterates in Pisces achieves the following ergodic convergence rate*

$$\frac{1}{T}\sum_{t=0}^{T-1} \left\|\nabla f(w^t)\right\|^2 \leq \frac{2\left(f(w^0) - f^*\right)}{\alpha(Q)T} + \frac{L}{2}\frac{\beta(Q)}{\alpha(Q)}\sigma_\ell^2 \qquad (4)$$
$$+3L^2 Q\beta(Q)\left(b^2+1\right)\left(\sigma_\ell^2 + \sigma_g^2 + G\right).$$

Guarantees convergence

Pisces: guided async FL with controlled staleness



① **Hard limit** on staleness via pace control at model aggregation

- Achieved by a neat yet provably effective algorithm

② **Soft limit** on staleness via informed participant selection

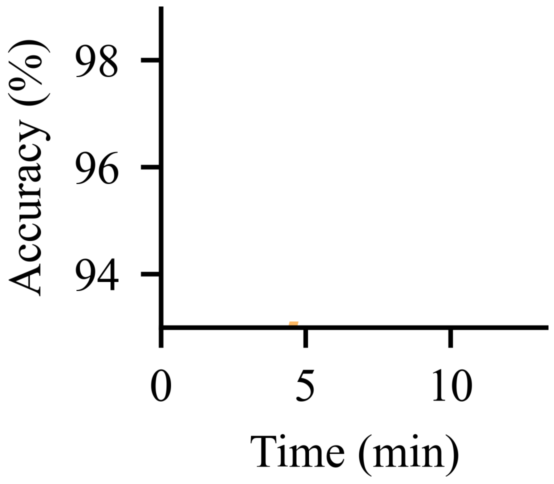
- Clients with higher score are selected more
- Definition of score U_i for client i:

$$U_i = \underbrace{\frac{1}{(\tilde{\tau}_i + 1)^{\beta}}}_{\text{Potential of low staleness}} \times \underbrace{|B_i| \sqrt{\frac{1}{|B_i|} \sum_{k \in B_i} Loss(k)^2}}_{\text{Data quality}}$$

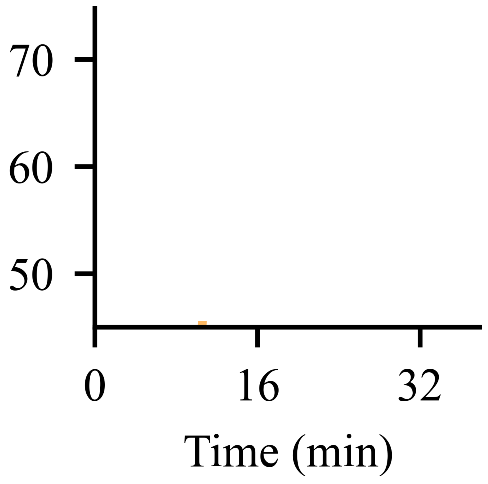
Pisces: guided async FL with controlled staleness

End-to-end efficiency

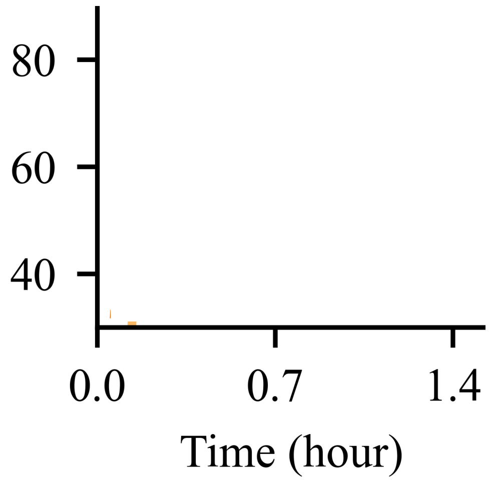
① Time-to-accuracy



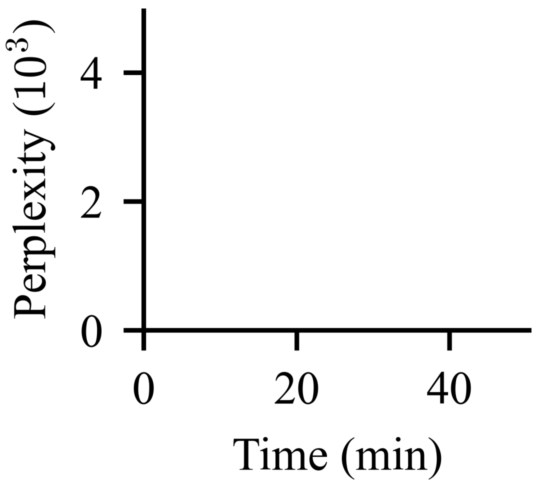
MNIST



FEMNIST



CIFAR-10



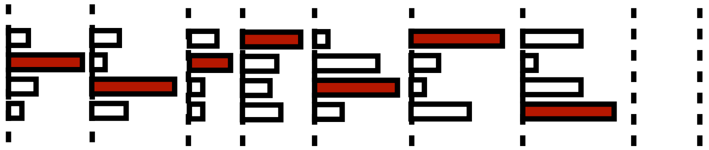
StackOverflow

Pisces: guided async FL with controlled staleness

Major competitors

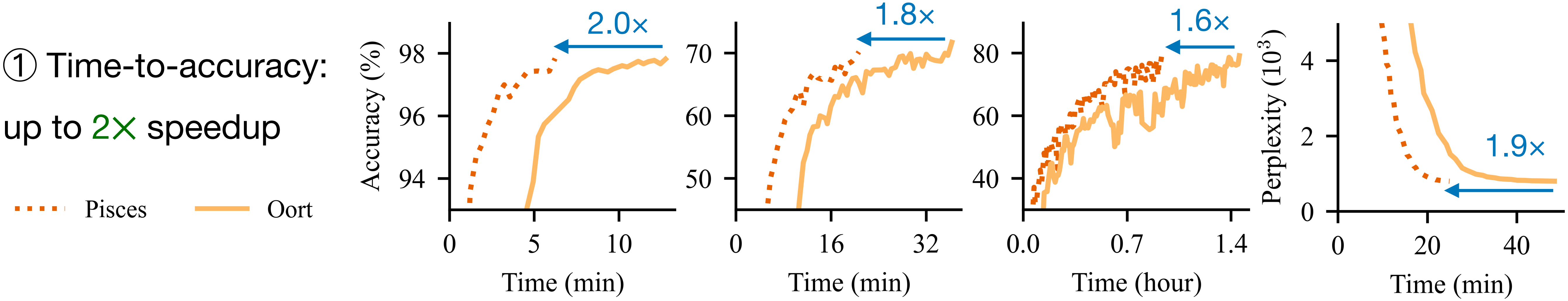
Oort (OSDI '21)

- SOTA sync FL
- Coupling speed and data quality



End-to-end efficiency

① Time-to-accuracy: up to 2× speedup



MNIST FEMNIST CIFAR-10 StackOverflow

[1] Oort: Efficient federated learning via guided participant selection

Pisces: guided async FL with controlled staleness

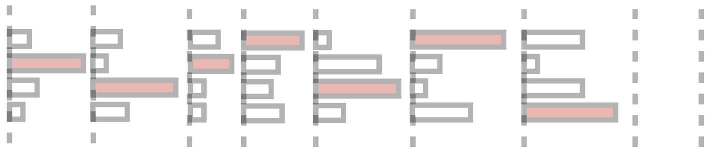
Major competitors

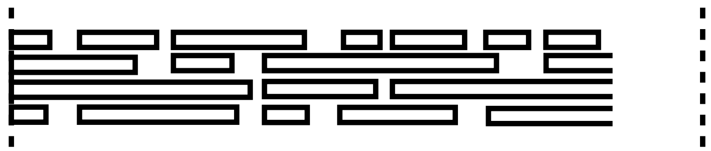
Oort (OSDI ’21)

- SOTA sync FL
- Coupling speed and data quality

FedBuff[1] (AISTATS ’22)

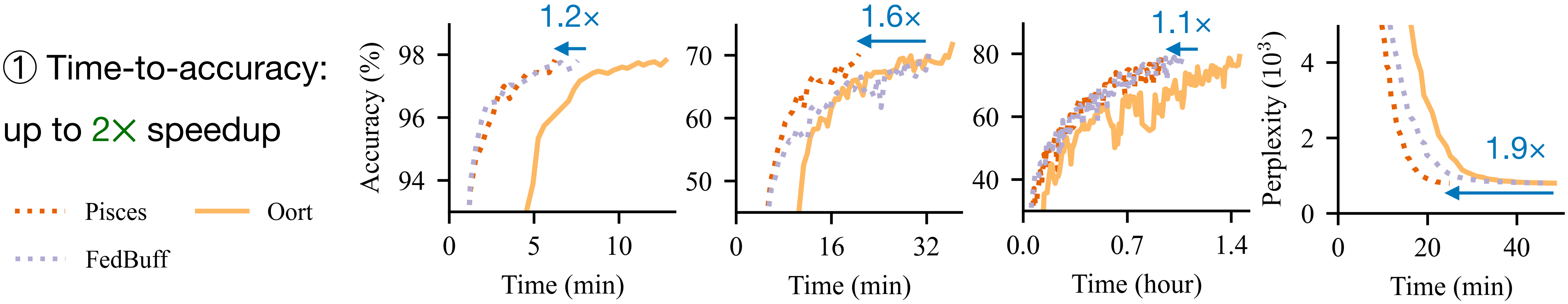
- SOTA async FL
- No bounded staleness
- No preference on data quality





End-to-end efficiency

① Time-to-accuracy: up to 2× speedup



MNIST FEMNIST CIFAR-10 StackOverflow

[1] Federated learning with buffered asynchronous aggregation

Pisces: guided async FL with controlled staleness

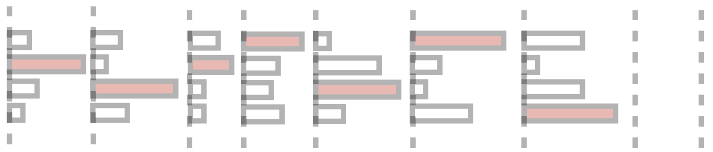
Major competitors

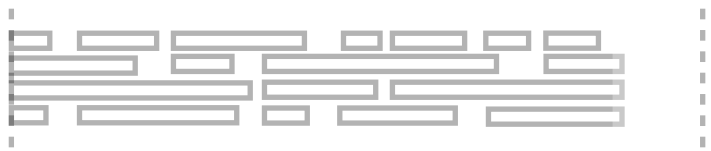
Oort (OSDI '21)

- SOTA sync FL
- Coupling speed and data quality

FedBuff[1] (AISTATS '22)

- SOTA async FL
- No bounded staleness
- No preference on data quality

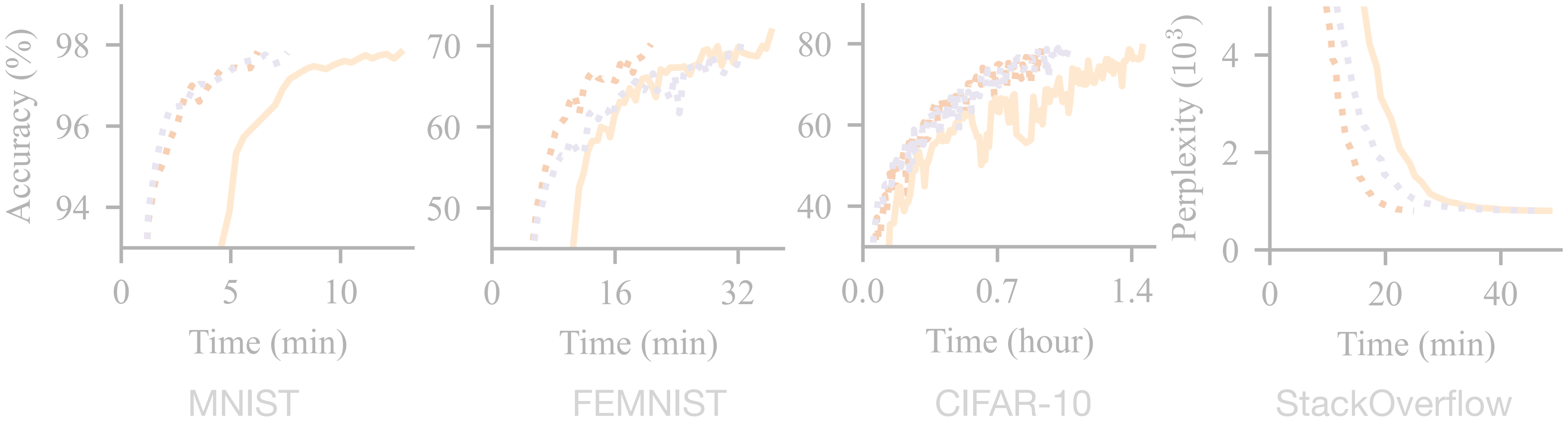




End-to-end efficiency

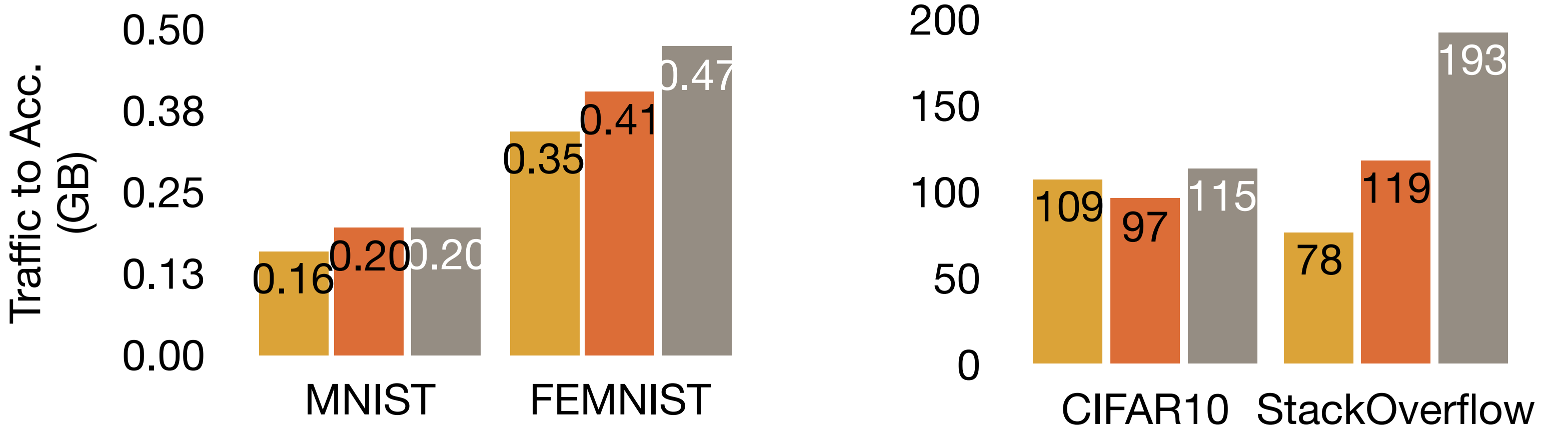
① Time-to-accuracy:

up to 2× speedup



② Traffic-to-accuracy:

No extra or even less

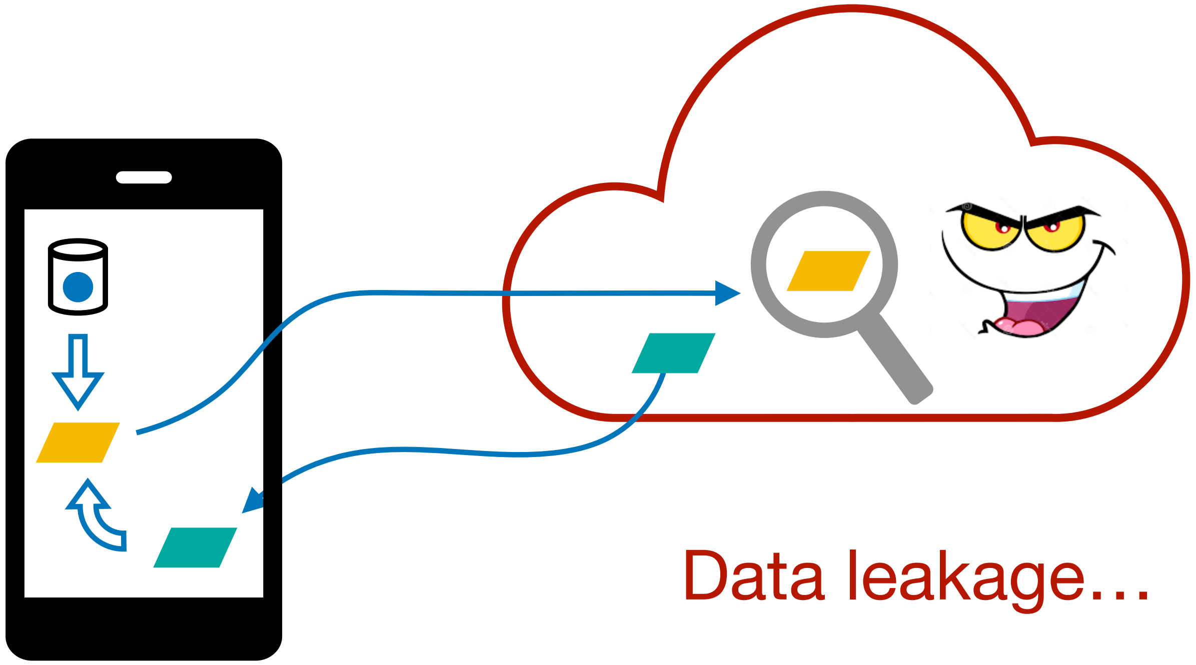


Pisces: guided async FL with eliminated staleness

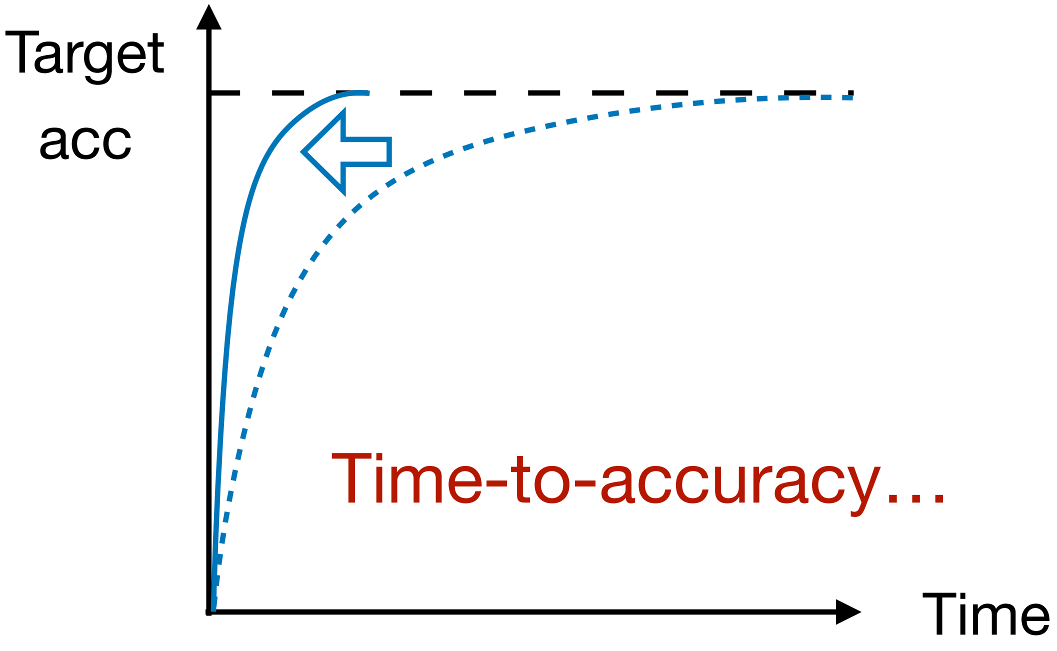
To boost efficiency in the presence of stragglers, the demands for clients' speed and data quality can be decoupled, with staleness carefully eliminated.

SoCC ‘22

My Work: build private and efficient cross-device FL



Data leakage…



Time-to-accuracy…

Weak privacy attackers

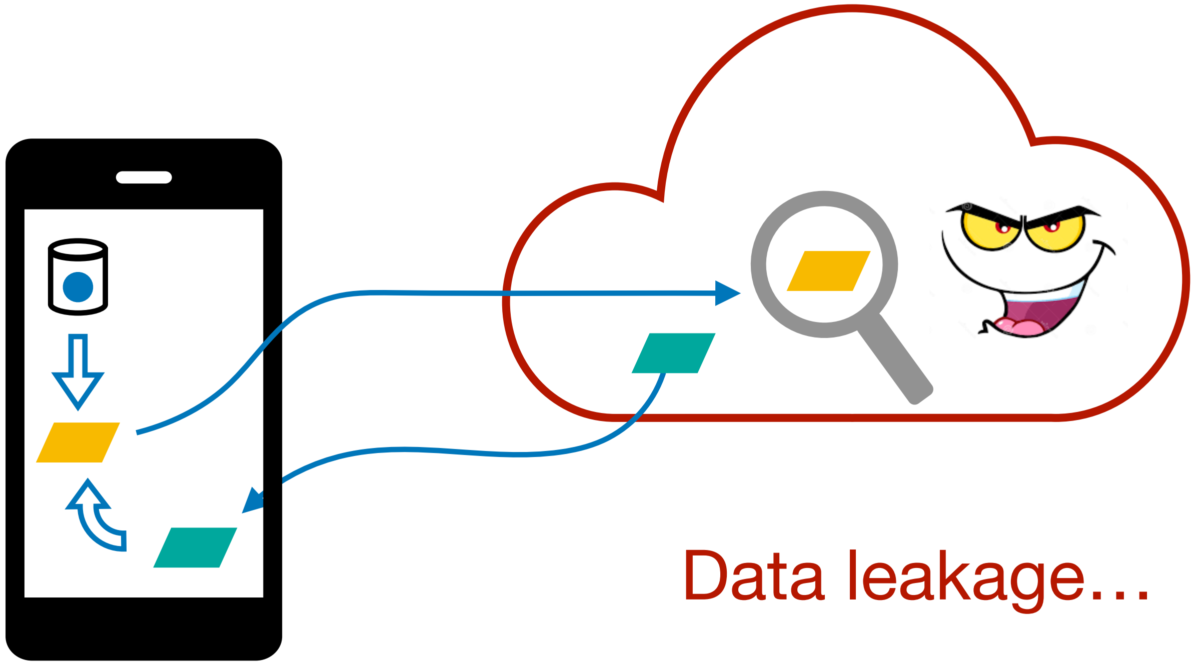
Efficient asynchronous training (SoCC ’22)

Dropout-resilient & pipeline-accelerated distributed differential privacy (EuroSys ’24)

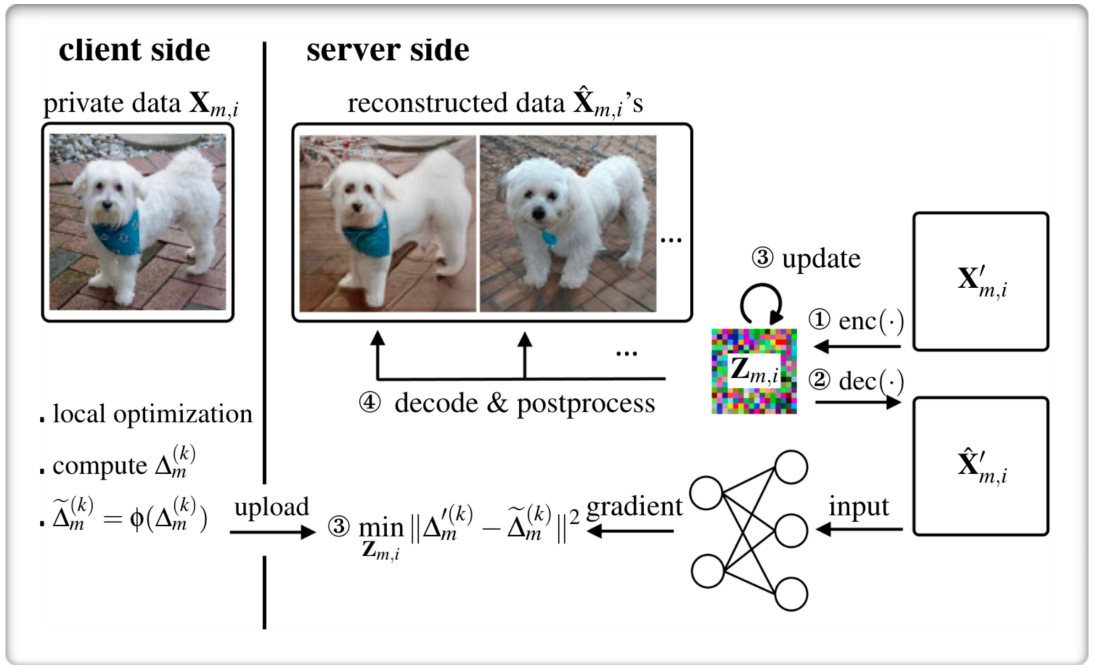
Secure participant selection (Security ’24)

Strong privacy attackers

The need for distributed differential privacy



Data leakage…


client side
server side
private data $\mathbf{X}_{m,i}$
reconstructed data $\hat{\mathbf{X}}_{m,i}$'s
③ update
$\mathbf{X}'_{m,i}$
① enc(·)
$\mathbf{Z}_{m,i}$
② dec(·)
④ decode & postprocess
$\hat{\mathbf{X}}'_{m,i}$
. local optimization
. compute $\Delta_m^{(k)}$
. $\widetilde{\Delta}_m^{(k)} = \phi(\Delta_m^{(k)})$
upload
③ $\min_{\mathbf{Z}_{m,i}} \|\Delta_m'^{(k)} - \widetilde{\Delta}_m^{(k)}\|^2$
gradient
input

e.g., data reconstruction[1] (Security '23)

[1] Gradient Obfuscation Gives a False Sense of Security in Federated Learning

The need for distributed differential privacy

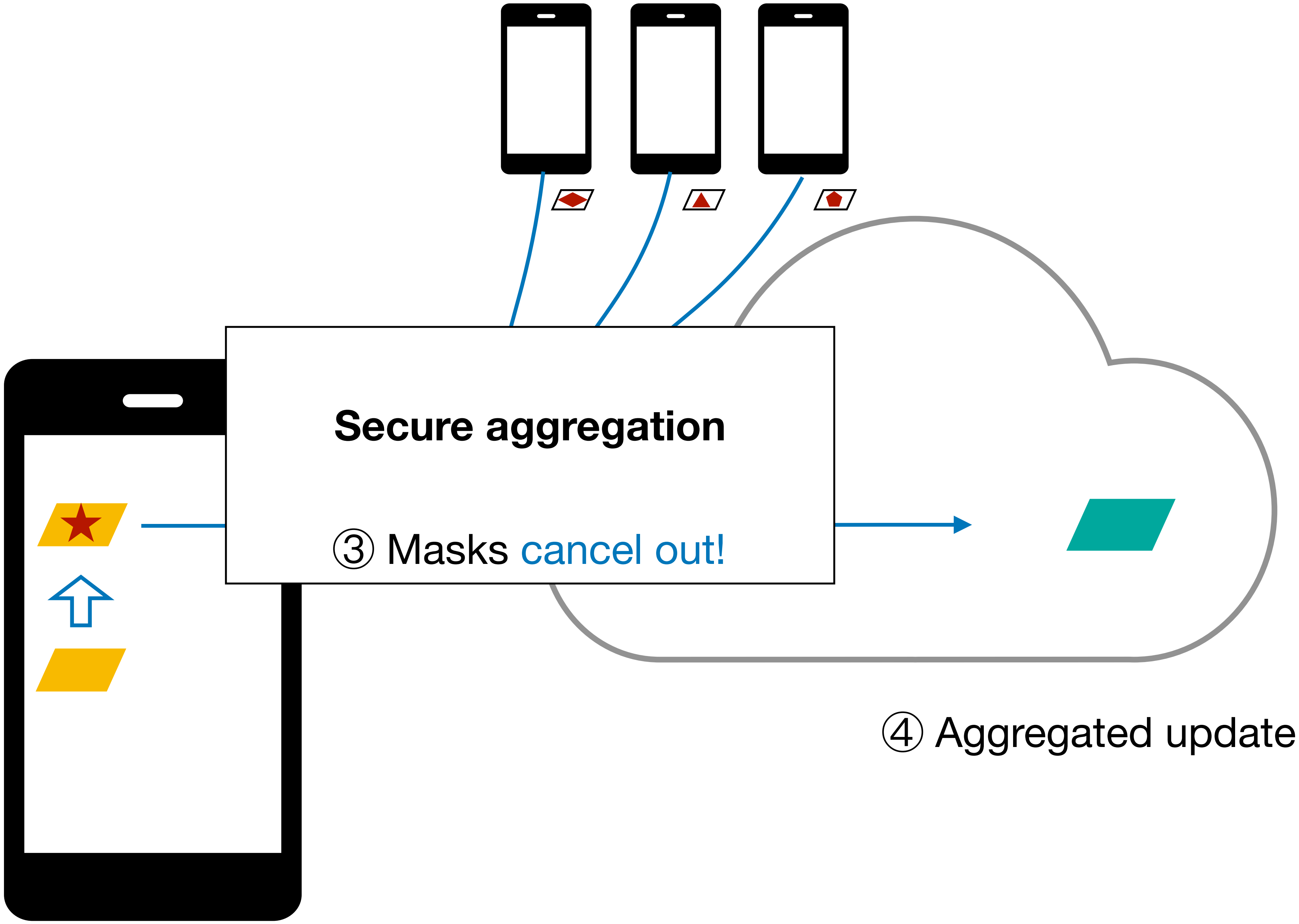
To conceal local updates?

Secure aggregation[1,2]
(CCS ’17, ‘20)

[1] Practical secure aggregation for privacy-preserving machine learning
[2] Secure Single-Server Aggregation with (Poly) Logarithmic Overhead

The need for distributed differential privacy

To conceal local updates?



② Masked local update

① Local update

Secure aggregation

③ Masks cancel out!

④ Aggregated update

The need for distributed differential privacy

To also perturb the aggregated update?

Differential Privacy[1]

[1] Calibrating Noise to Sensitivity in Privacy Data Analysis

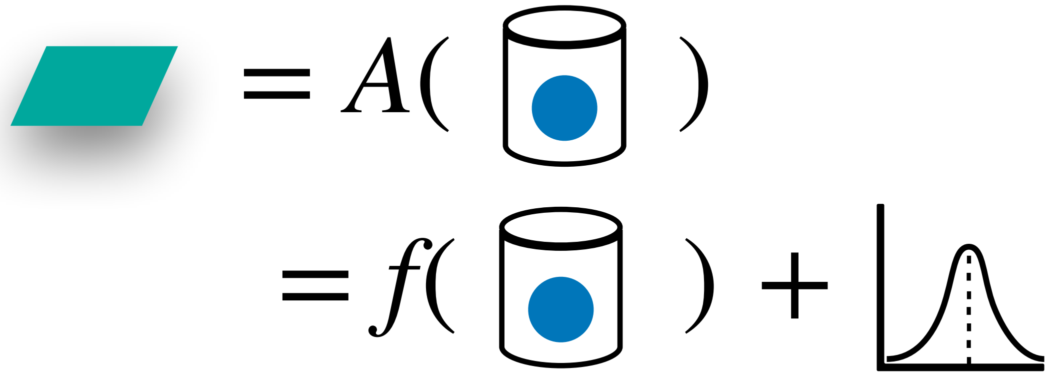
The need for distributed differential privacy

To also perturb the aggregated update?

Differential Privacy[1]

Sacrifice the precision

noisy aggregated update



aggregation

local updates

random noise

For enhanced privacy

DP ensures that be **insensitive** to the impact of any single local update in

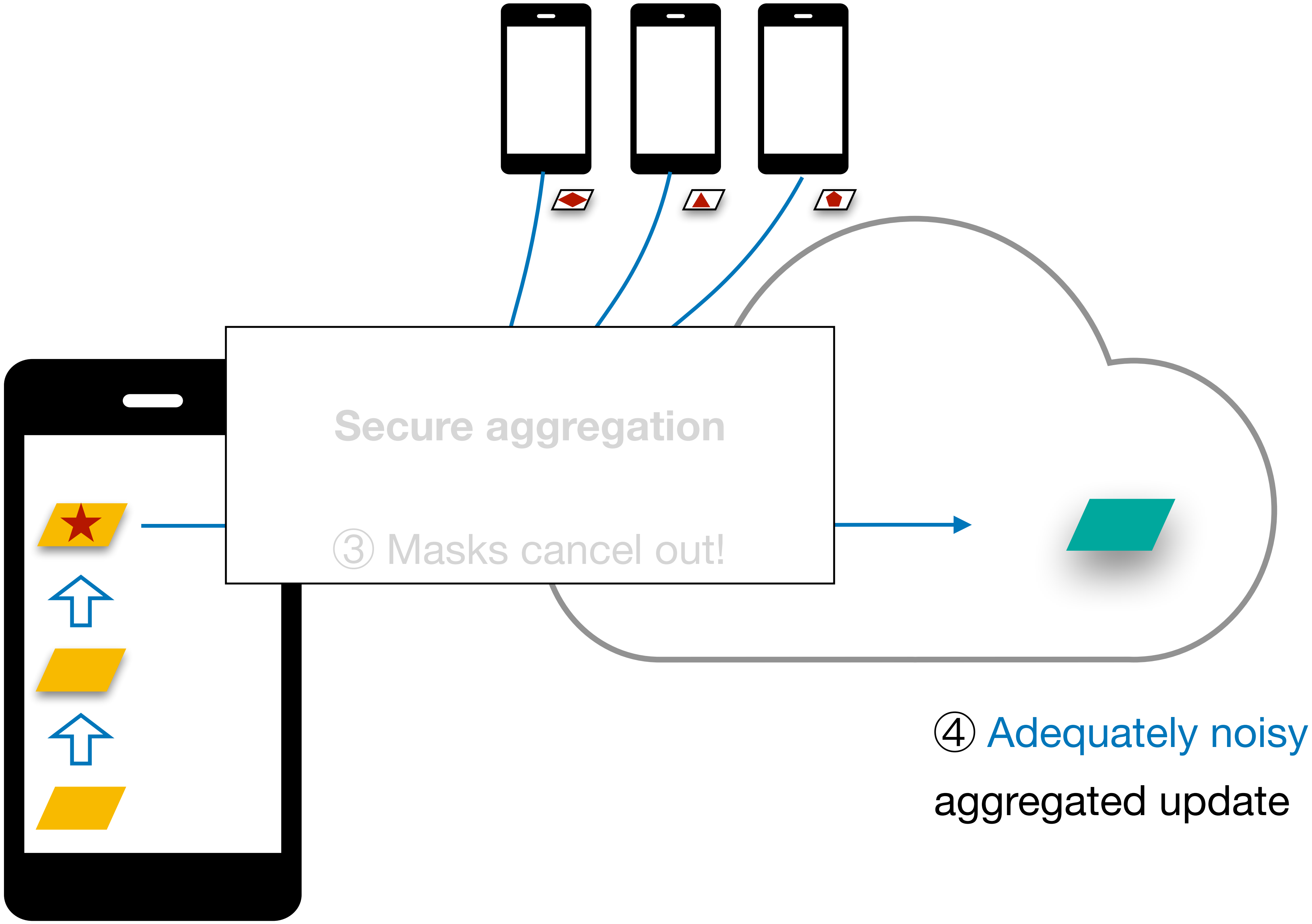
[1] Calibrating Noise to Sensitivity in Privacy Data Analysis

The need for distributed differential privacy

To also perturb the aggregated update?

② Masked slightly noisy local update

① Slightly noisy local update



Secure aggregation

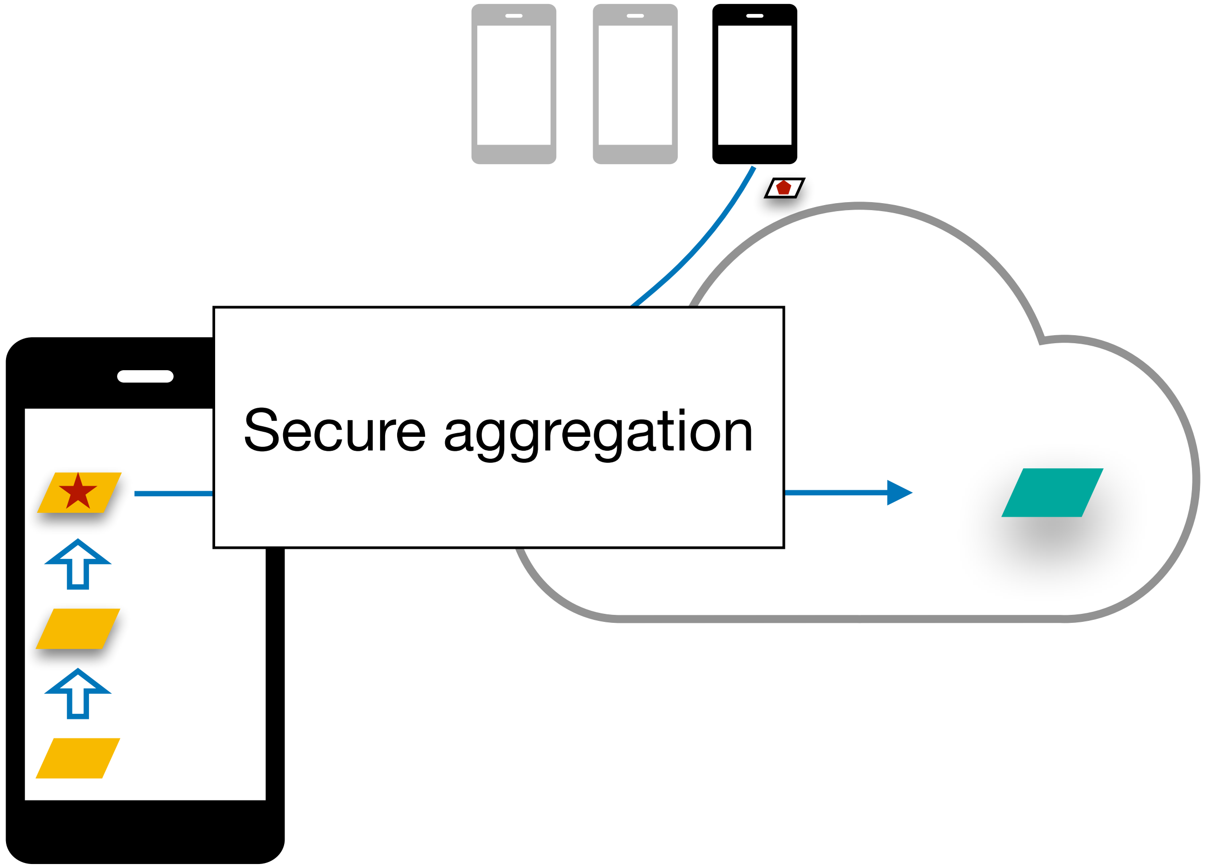
③ Masks cancel out!

④ Adequately noisy aggregated update

⓪ Global privacy budget ϵ → Calculate the minimum required noise for each round

Three practical issues in distributed DP

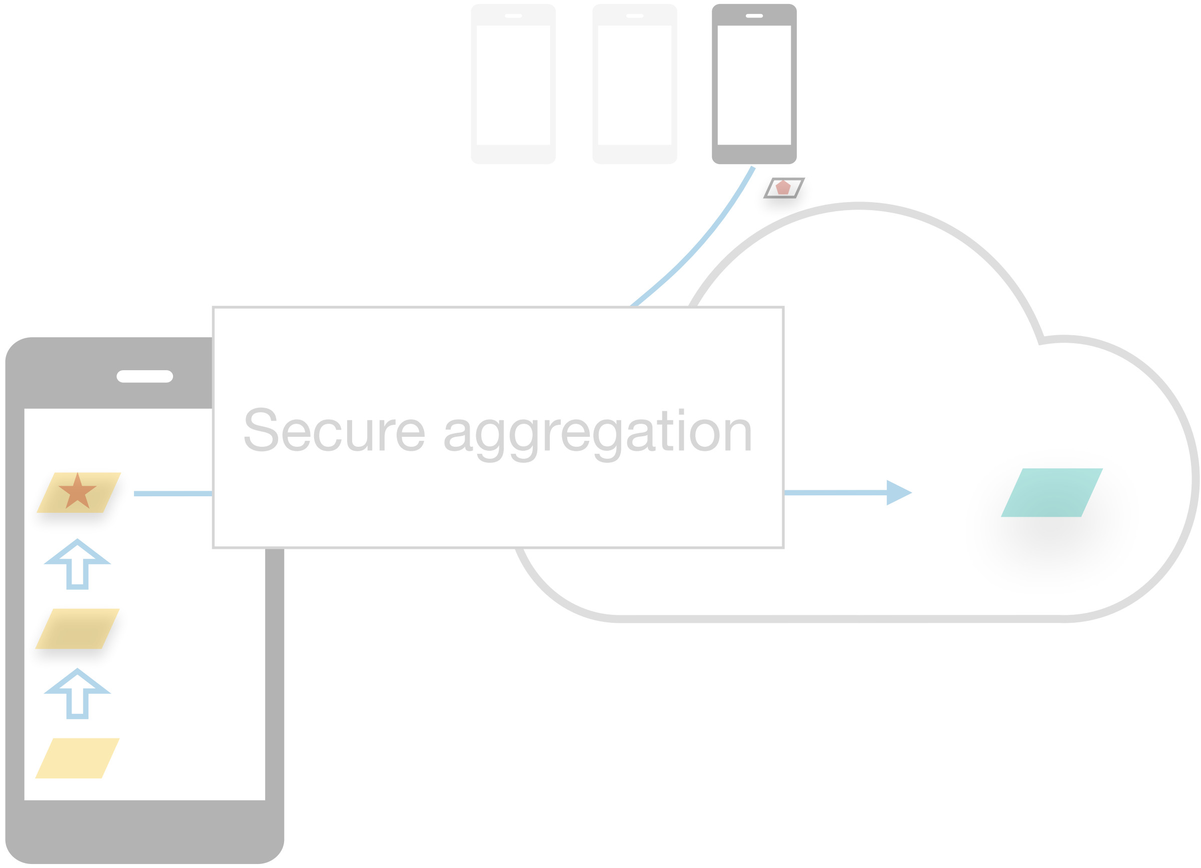
1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime

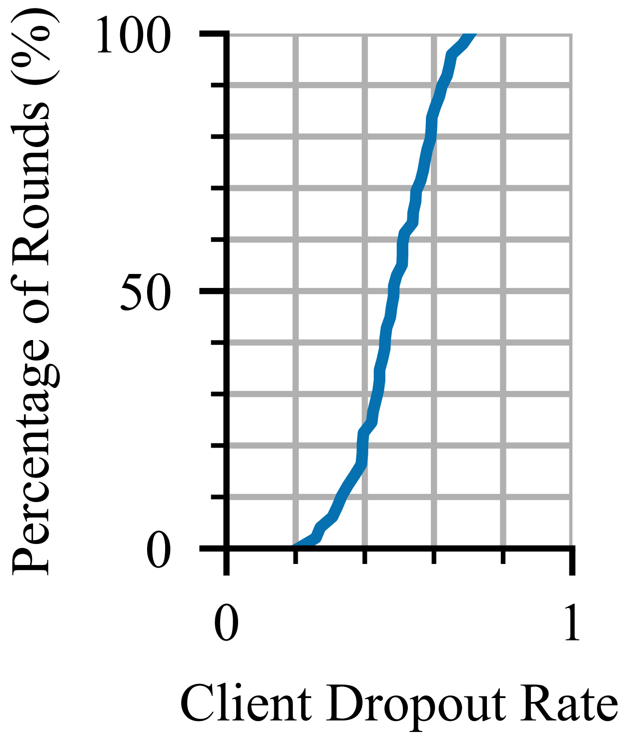


Secure aggregation

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime





Client behaviors simulated with 100 volatile users from the FLASH dataset[1] (WWW '21)

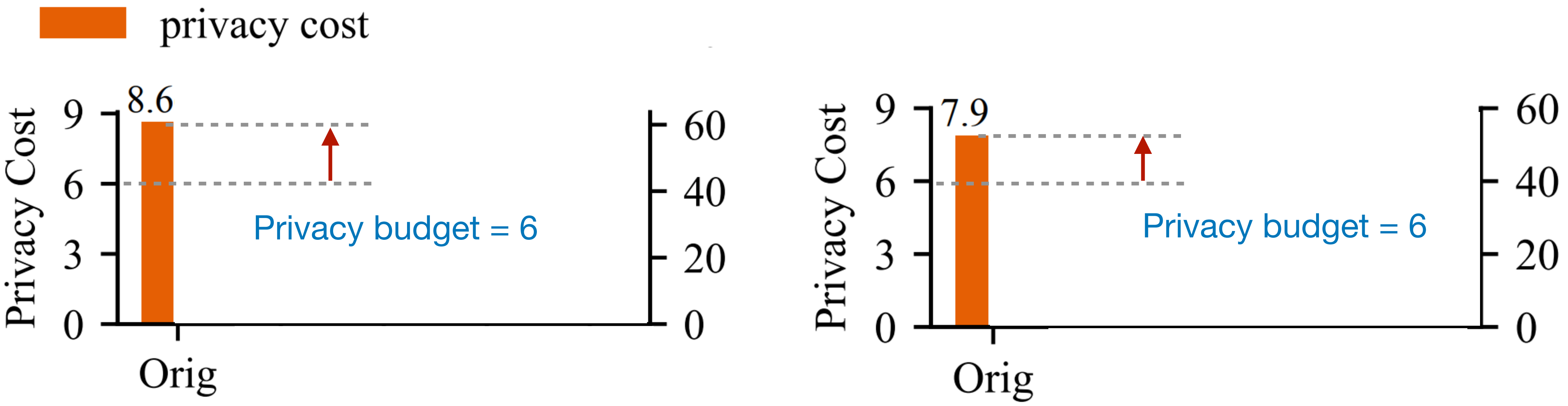
[1] Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime
   - Insufficient noise for target privacy

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime
   - Insufficient noise for target privacy



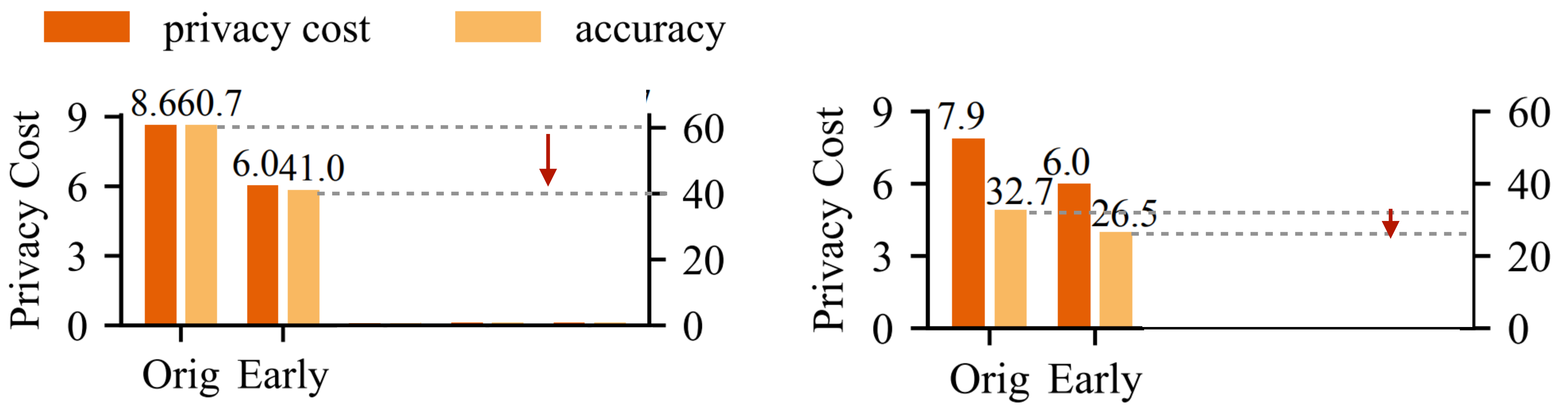
CIFAR-10
Testbed

CIFAR-100
Testbed

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime
   - Insufficient noise for target privacy
   - Naive solutions and their limitations
     - **Early**: early stop when budget runs out—hurts utility

Privacy budget = 6



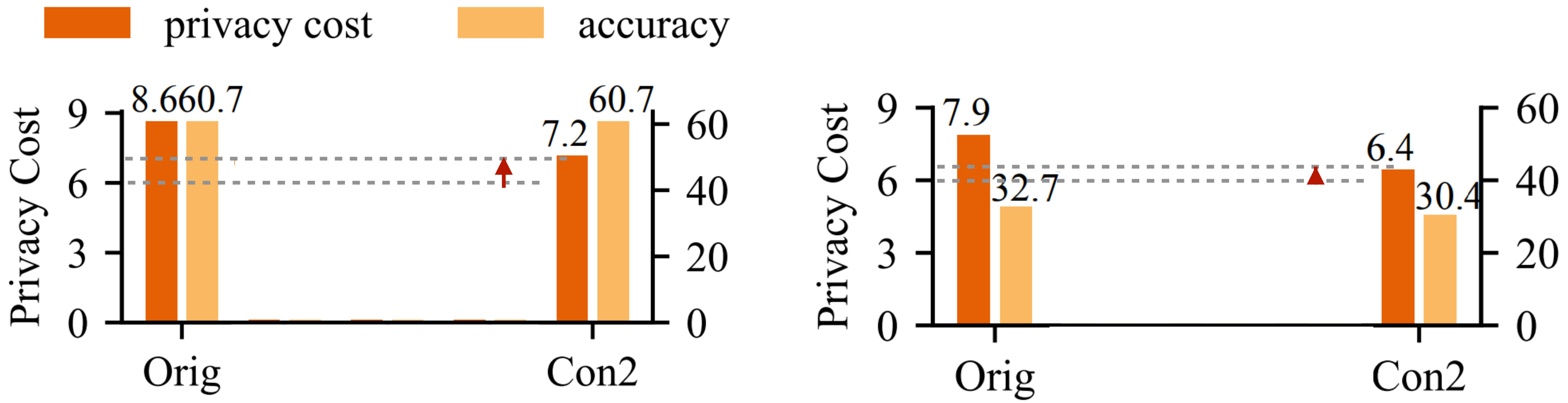
CIFAR-10
Testbed

CIFAR-100
Testbed

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime
   - Insufficient noise for target privacy
   - Naive solutions and their limitations
     - **Early**: early stop when budget runs out—hurts utility
     - **Con**: proactively add more noise—requires expertise

Privacy budget = 6



CIFAR-10

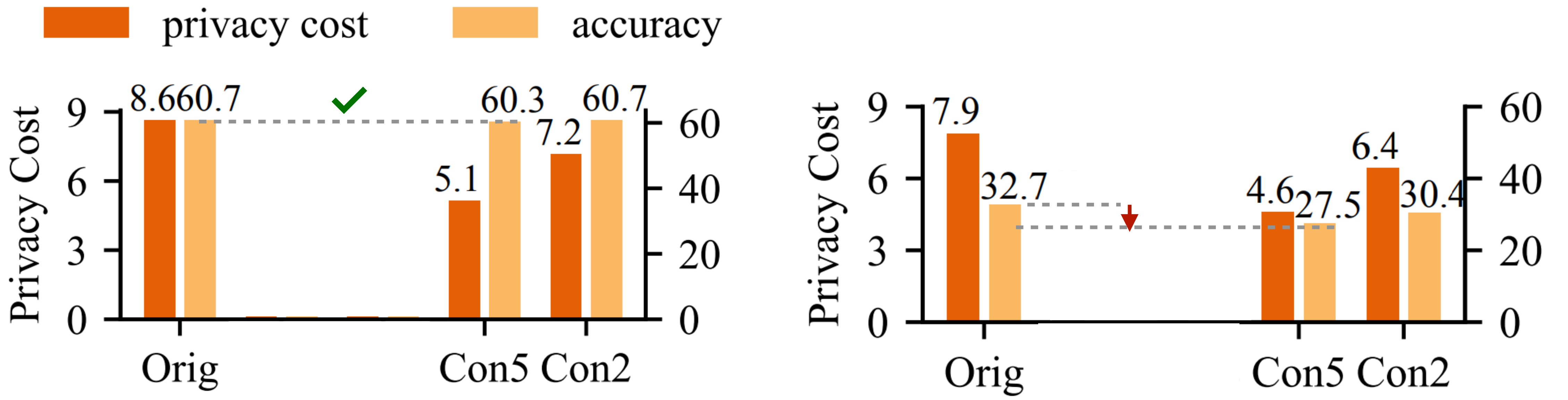
CIFAR-100

Too optimistic: privacy compromised

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
   - Client dropout can occur anytime
   - Insufficient noise for target privacy
   - Naive solutions and their limitations
     - **Early**: early stop when budget runs out—hurts utility
     - **Con**: proactively add more noise—requires expertise

Privacy budget = 6



CIFAR-10

CIFAR-100

Too pessimistic: utility may or may not suffer

Dropout-resilient noise enforcement

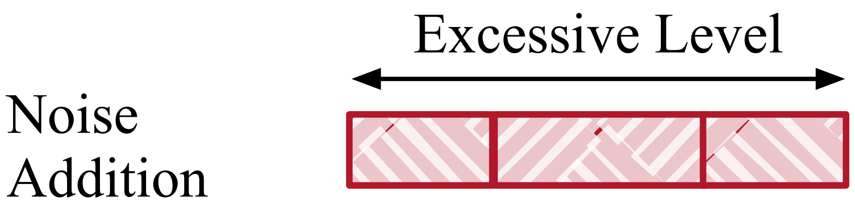
Goal: achieve the best privacy-utility tradeoff without domain knowledge

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components

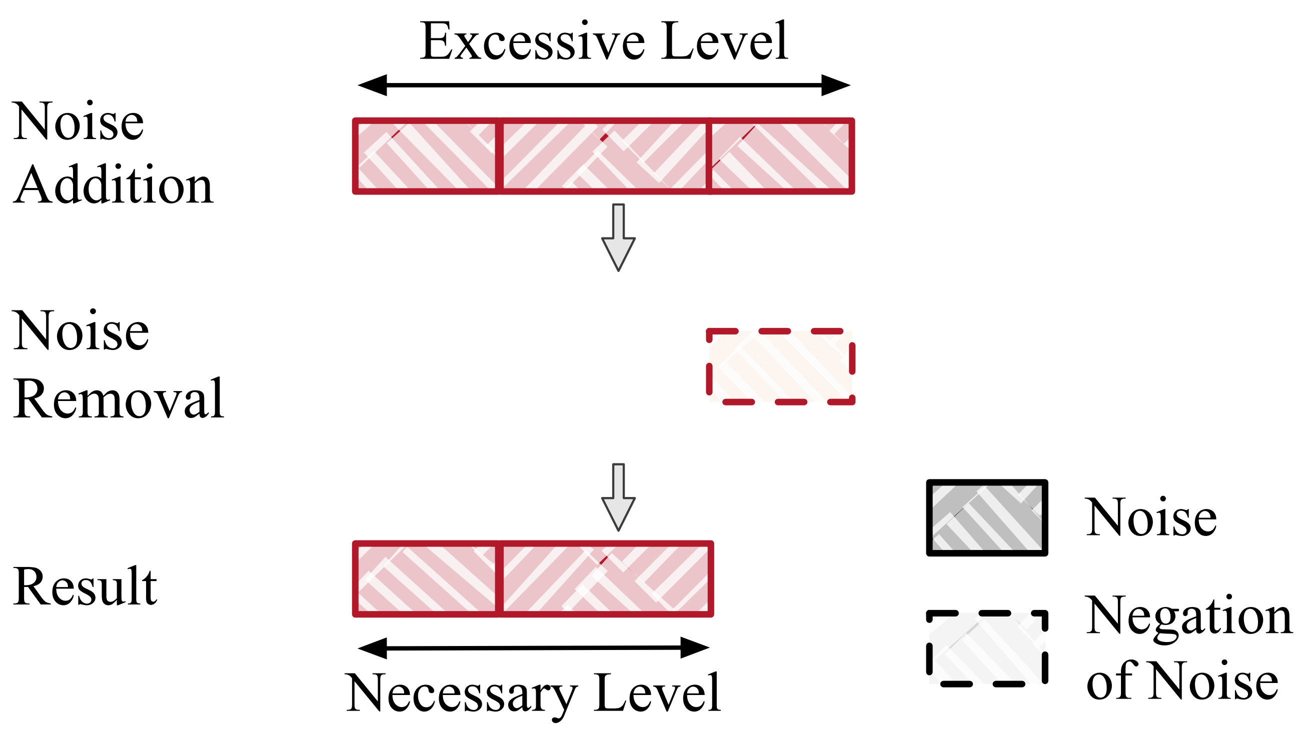


Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server



Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

Concrete example

Sampled clients $|S| = 4$

Minimum necessary noise level $\sigma_*^2 = 1$

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

Concrete example

Each client adds noise $n_i \sim \chi(1/2)$
to tolerate up to 2 clients to drop

Sampled clients $|S| = 4$

Add Dropout tolerance $t = 2,$

Minimum necessary noise level $\sigma_*^2 = 1$

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

Concrete example

Sampled clients $|S| = 4$

Add Dropout tolerance $t = 2,$

Minimum necessary noise level $\sigma_*^2 = 1$

Clients

Each client adds noise $n_i \sim \chi(1/2)$ to tolerate up to 2 clients to drop

$n_{1,0} \sim \chi(1/4)$ $n_{1,1} \sim \chi(1/12)$ $n_{1,2} \sim \chi(1/6)$

$n_{2,0} \sim \chi(1/4)$ $n_{2,1} \sim \chi(1/12)$ $n_{2,2} \sim \chi(1/6)$

$n_{3,0} \sim \chi(1/4)$ $n_{3,1} \sim \chi(1/12)$ $n_{3,2} \sim \chi(1/6)$

$n_{4,0} \sim \chi(1/4)$ $n_{4,1} \sim \chi(1/12)$ $n_{4,2} \sim \chi(1/6)$

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

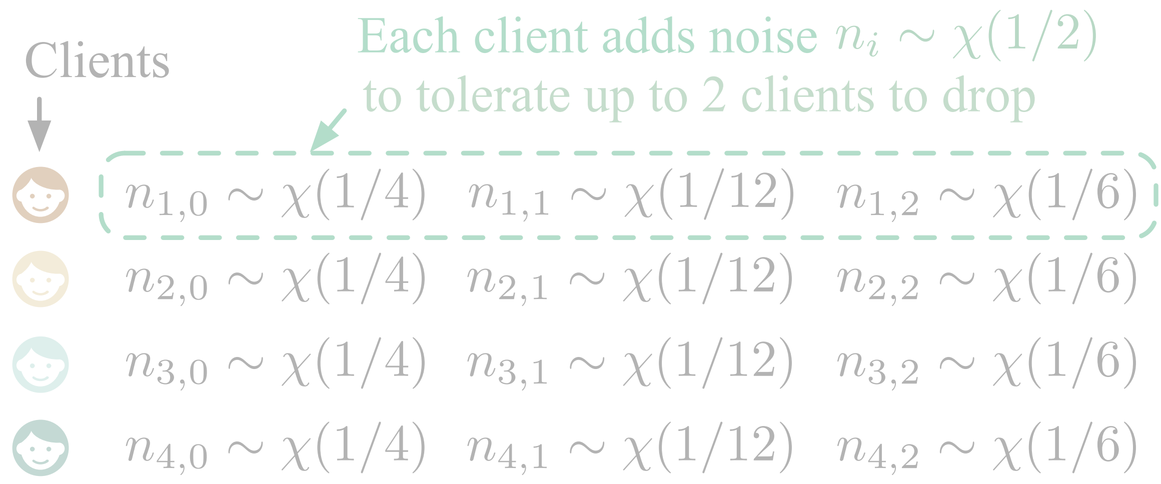
Concrete example

Add

Sampled clients $|S| = 4$

Dropout tolerance $t = 2$,

Minimum necessary noise level $\sigma_*^2 = 1$



Clients

Each client adds noise $n_i \sim \chi(1/2)$ to tolerate up to 2 clients to drop

$n_{1,0} \sim \chi(1/4)$ $n_{1,1} \sim \chi(1/12)$ $n_{1,2} \sim \chi(1/6)$

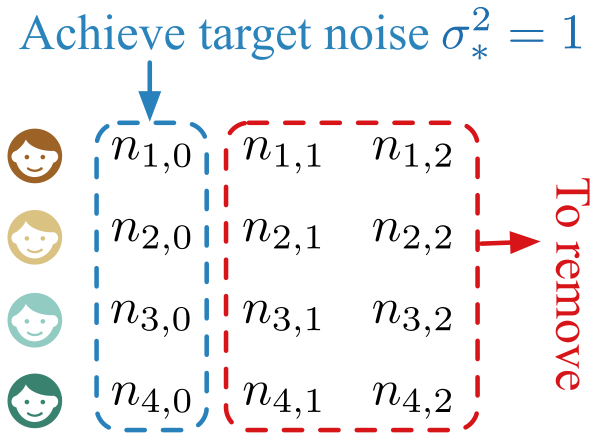
$n_{2,0} \sim \chi(1/4)$ $n_{2,1} \sim \chi(1/12)$ $n_{2,2} \sim \chi(1/6)$

$n_{3,0} \sim \chi(1/4)$ $n_{3,1} \sim \chi(1/12)$ $n_{3,2} \sim \chi(1/6)$

$n_{4,0} \sim \chi(1/4)$ $n_{4,1} \sim \chi(1/12)$ $n_{4,2} \sim \chi(1/6)$

If 0 client drops

Then remove



Achieve target noise $\sigma_*^2 = 1$

$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

$n_{3,0}$ $n_{3,1}$ $n_{3,2}$

$n_{4,0}$ $n_{4,1}$ $n_{4,2}$

To remove

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

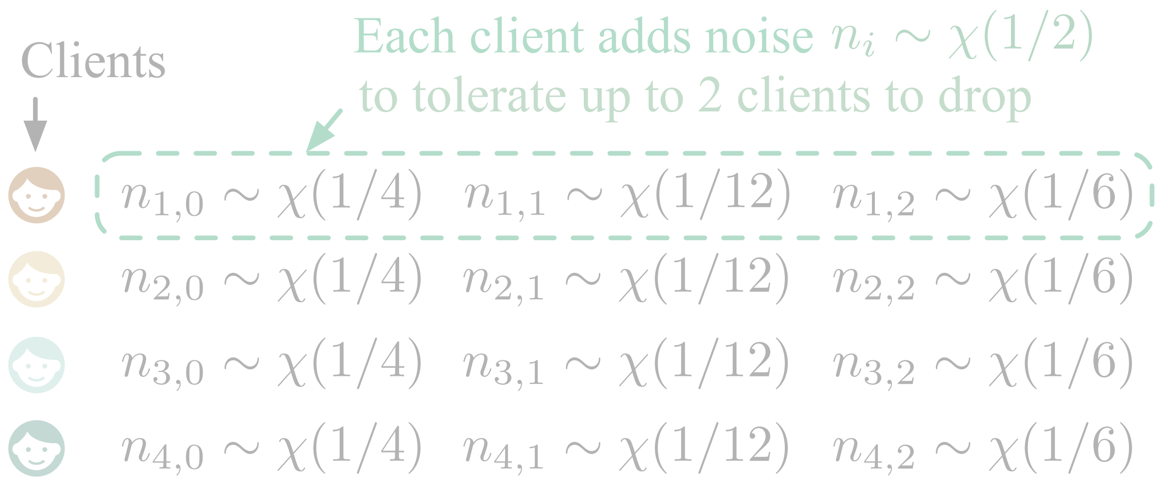
Concrete example

Add

Sampled clients $|S| = 4$

Dropout tolerance $t = 2$,

Minimum necessary noise level $\sigma_*^2 = 1$



Clients

Each client adds noise $n_i \sim \chi(1/2)$ to tolerate up to 2 clients to drop

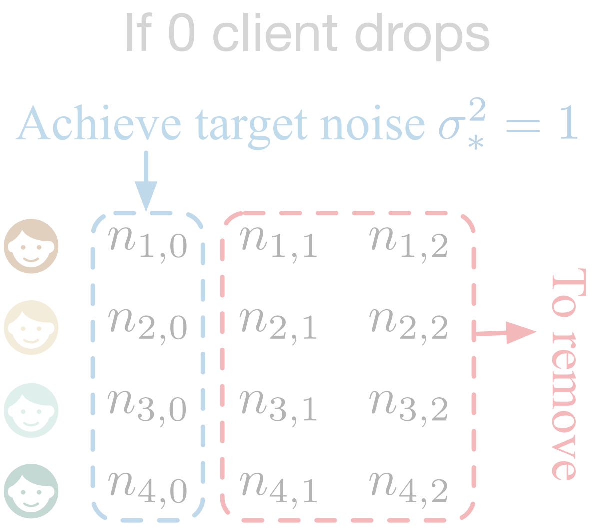
$n_{1,0} \sim \chi(1/4)$ $n_{1,1} \sim \chi(1/12)$ $n_{1,2} \sim \chi(1/6)$

$n_{2,0} \sim \chi(1/4)$ $n_{2,1} \sim \chi(1/12)$ $n_{2,2} \sim \chi(1/6)$

$n_{3,0} \sim \chi(1/4)$ $n_{3,1} \sim \chi(1/12)$ $n_{3,2} \sim \chi(1/6)$

$n_{4,0} \sim \chi(1/4)$ $n_{4,1} \sim \chi(1/12)$ $n_{4,2} \sim \chi(1/6)$

Then remove



If 0 client drops

Achieve target noise $\sigma_*^2 = 1$

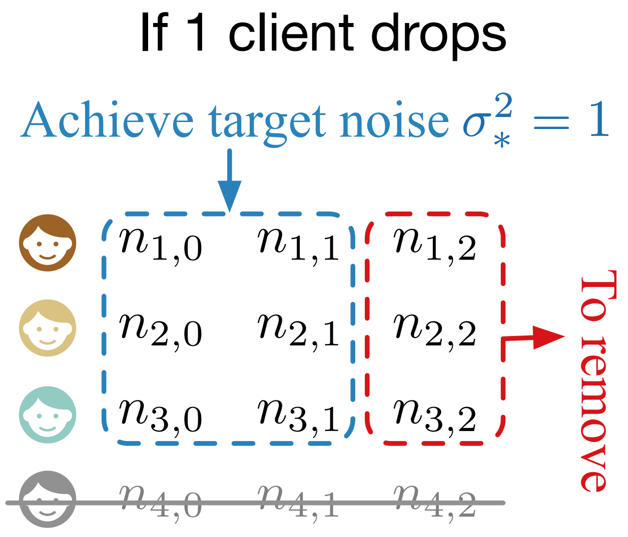
$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

$n_{3,0}$ $n_{3,1}$ $n_{3,2}$

$n_{4,0}$ $n_{4,1}$ $n_{4,2}$

To remove



If 1 client drops

Achieve target noise $\sigma_*^2 = 1$

$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

$n_{3,0}$ $n_{3,1}$ $n_{3,2}$

~~$n_{4,0}$ $n_{4,1}$ $n_{4,2}$~~

To remove

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

- Each client first adds excessive noise as separate components
- After aggregation, unnecessary ones are removed by the server

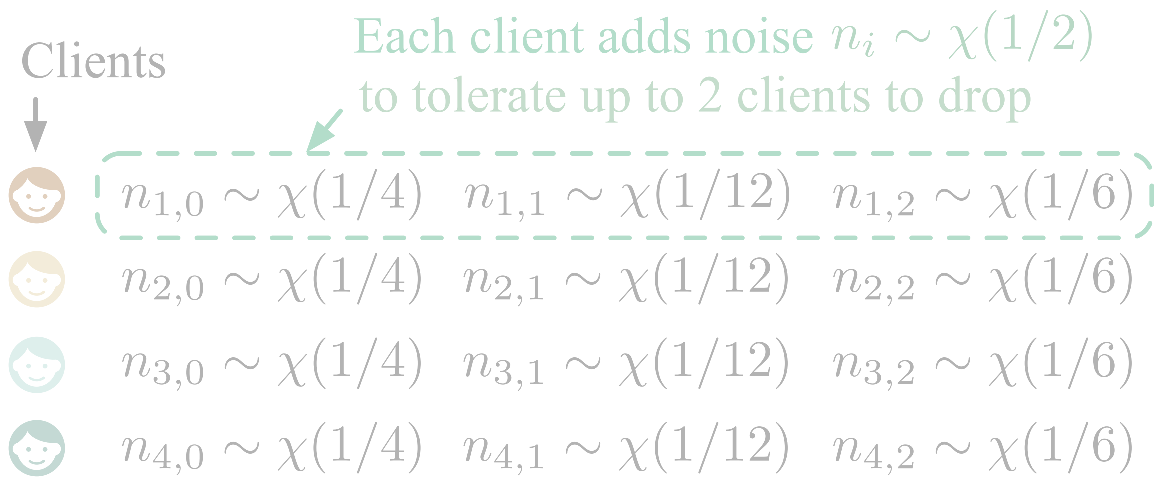
Concrete example

Add

Sampled clients $|S| = 4$

Dropout tolerance $t = 2$,

Minimum necessary noise level $\sigma_*^2 = 1$

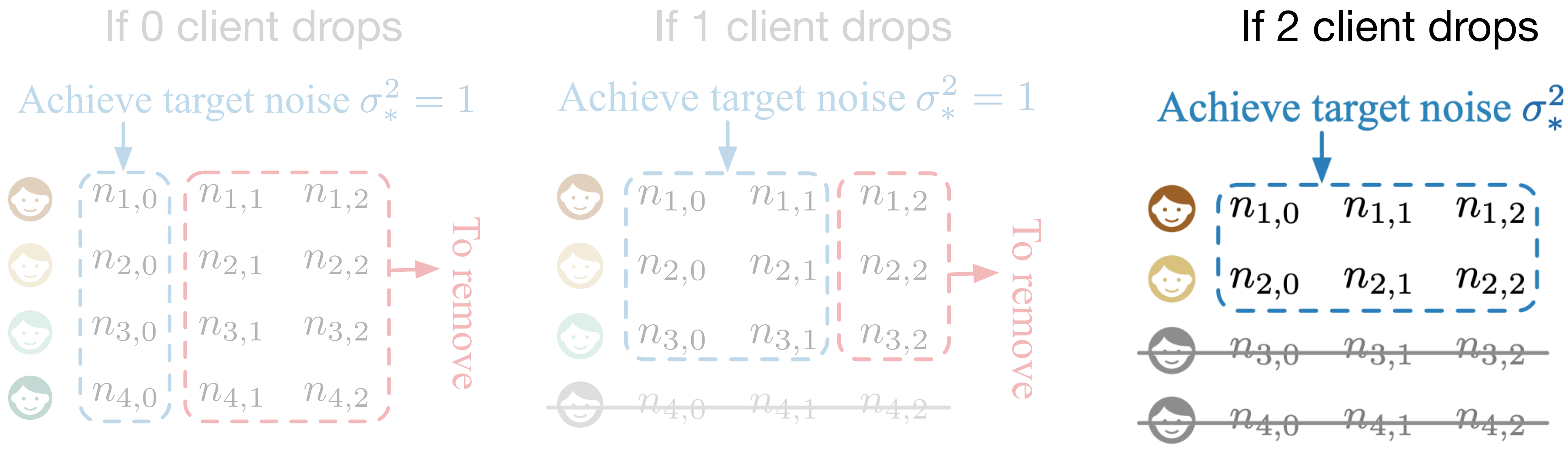


Clients

Each client adds noise $n_i \sim \chi(1/2)$ to tolerate up to 2 clients to drop

$n_{1,0} \sim \chi(1/4)$	$n_{1,1} \sim \chi(1/12)$	$n_{1,2} \sim \chi(1/6)$
$n_{2,0} \sim \chi(1/4)$	$n_{2,1} \sim \chi(1/12)$	$n_{2,2} \sim \chi(1/6)$
$n_{3,0} \sim \chi(1/4)$	$n_{3,1} \sim \chi(1/12)$	$n_{3,2} \sim \chi(1/6)$
$n_{4,0} \sim \chi(1/4)$	$n_{4,1} \sim \chi(1/12)$	$n_{4,2} \sim \chi(1/6)$

Then remove



If 0 client drops

Achieve target noise $\sigma_*^2 = 1$

$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

$n_{3,0}$ $n_{3,1}$ $n_{3,2}$

$n_{4,0}$ $n_{4,1}$ $n_{4,2}$

To remove

If 1 client drops

Achieve target noise $\sigma_*^2 = 1$

$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

$n_{3,0}$ $n_{3,1}$ $n_{3,2}$

~~$n_{4,0}$ $n_{4,1}$ $n_{4,2}$~~

To remove

If 2 client drops

Achieve target noise $\sigma_*^2 = 1$

$n_{1,0}$ $n_{1,1}$ $n_{1,2}$

$n_{2,0}$ $n_{2,1}$ $n_{2,2}$

~~$n_{3,0}$ $n_{3,1}$ $n_{3,2}$~~

~~$n_{4,0}$ $n_{4,1}$ $n_{4,2}$~~

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

Intuition: add-then-remove

Concrete example

Formal definition: **XNoise**

- Noise addition: decompose Client i's added noise $n_i \sim \chi\left(\frac{\sigma_*^2}{|S| - t}\right)$ into $t + 1$ components: $n_i = \sum_{k=0}^{t} n_{i,k}$, $n_{i,0} \sim \chi\left(\frac{\sigma_*^2}{|S|}\right)$, and $n_{i,k} \sim \chi\left(\frac{\sigma_*^2}{(|S| - k + 1)(|S| - k)}\right)$ $(k \in [t])$
- Noise removal: when there are $|D|$ clients dropping out, the noise components $n_{i,k}$ contributed by the surviving clients $i \in S \backslash D$ with the index $k > |D|$ becomes excessive and is removed by the server

Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

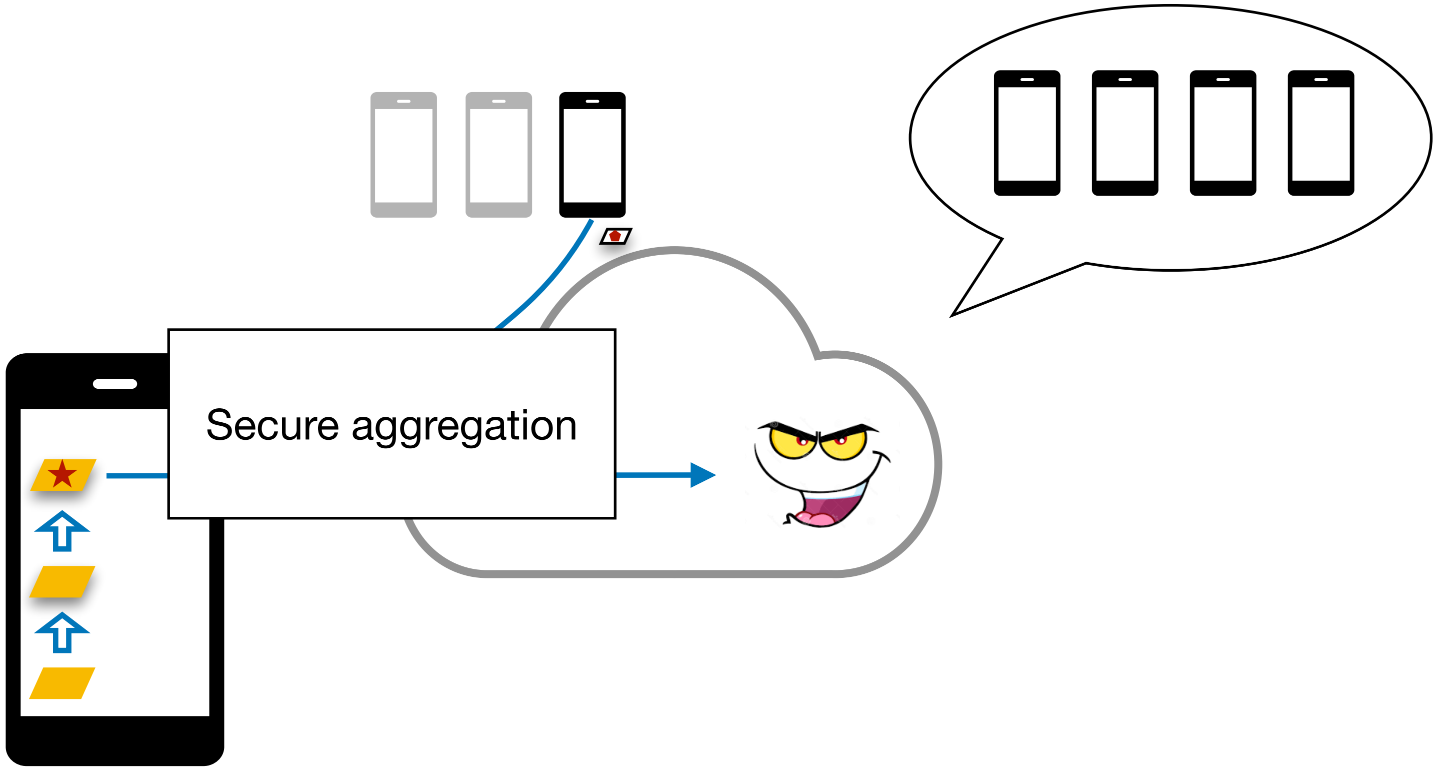
Intuition: add-then-remove

Concrete example

Formal definition: XNoise

Preventing adversarial server from understating dropout

- Mislead survivals to remove more noise than needed



Dropout-resilient noise enforcement

Goal: achieve the best privacy-utility tradeoff without domain knowledge

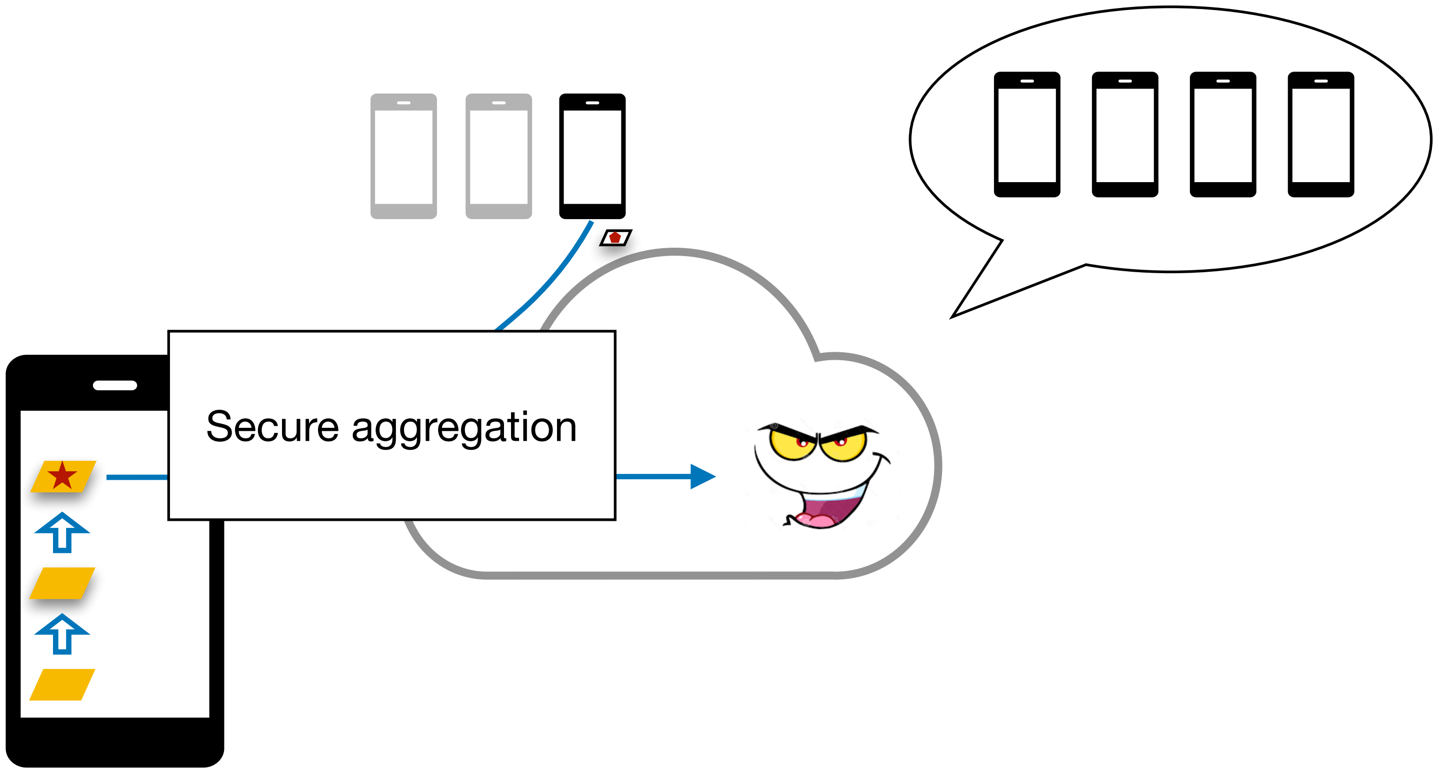
Intuition: add-then-remove

Concrete example

Formal definition: **XNoise**

Preventing adversarial server from understating dropout

- Mislead survivals to remove more noise than needed
- Enable verification via a secure signature scheme



Dropout-resilient noise enforcement

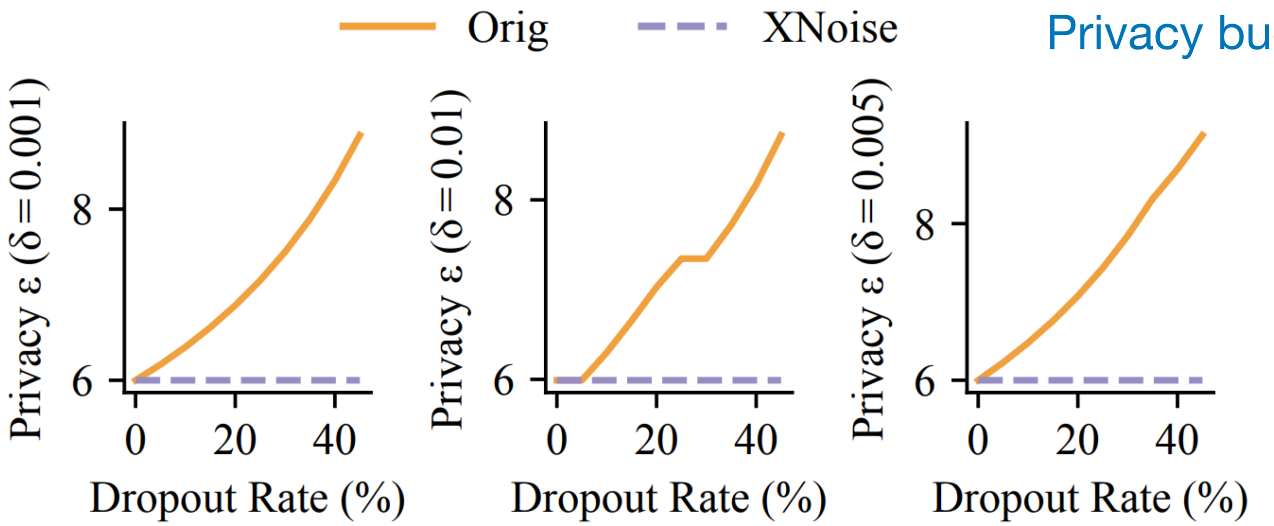
Effectiveness

Dropout-resilient noise enforcement

Effectiveness

- Improves privacy

Privacy budget = 6

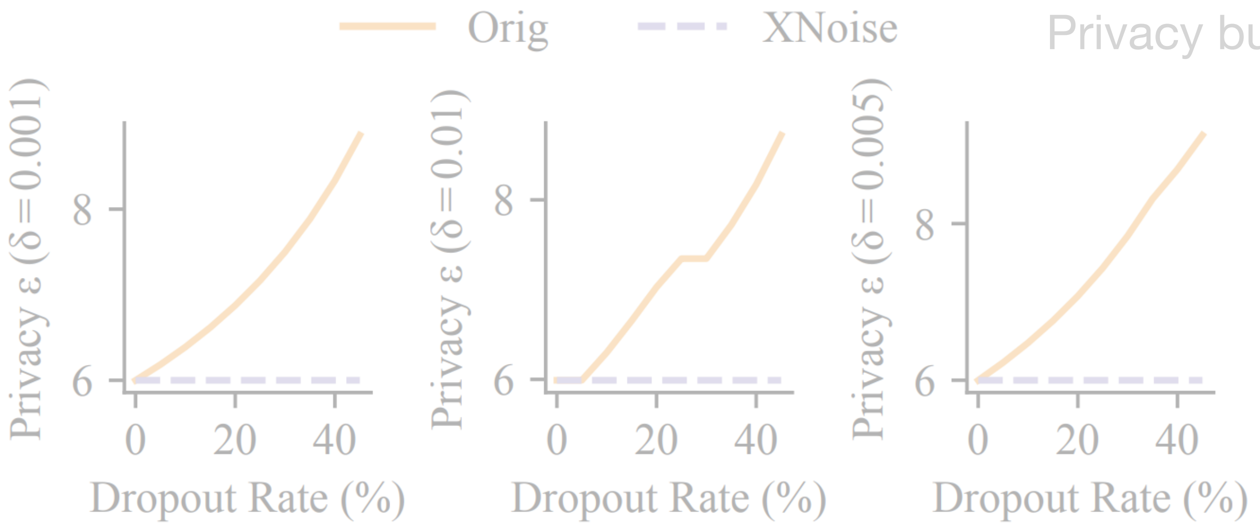


(a) FEMNIST. **(b)** CIFAR-10. **(c)** Reddit.

Dropout-resilient noise enforcement

Effectiveness

Improves privacy

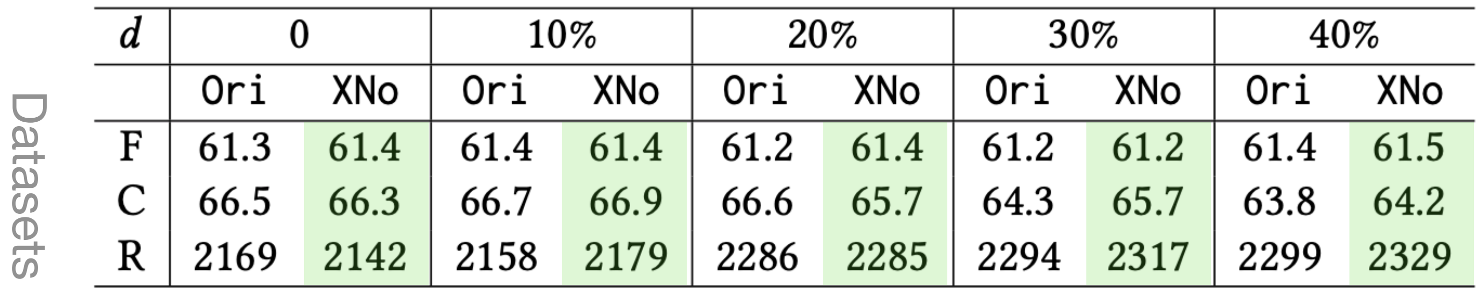


Privacy budget = 6

(a) FEMNIST. (b) CIFAR-10. (c) Reddit.

without sacrificing final model utility

Dropout rates



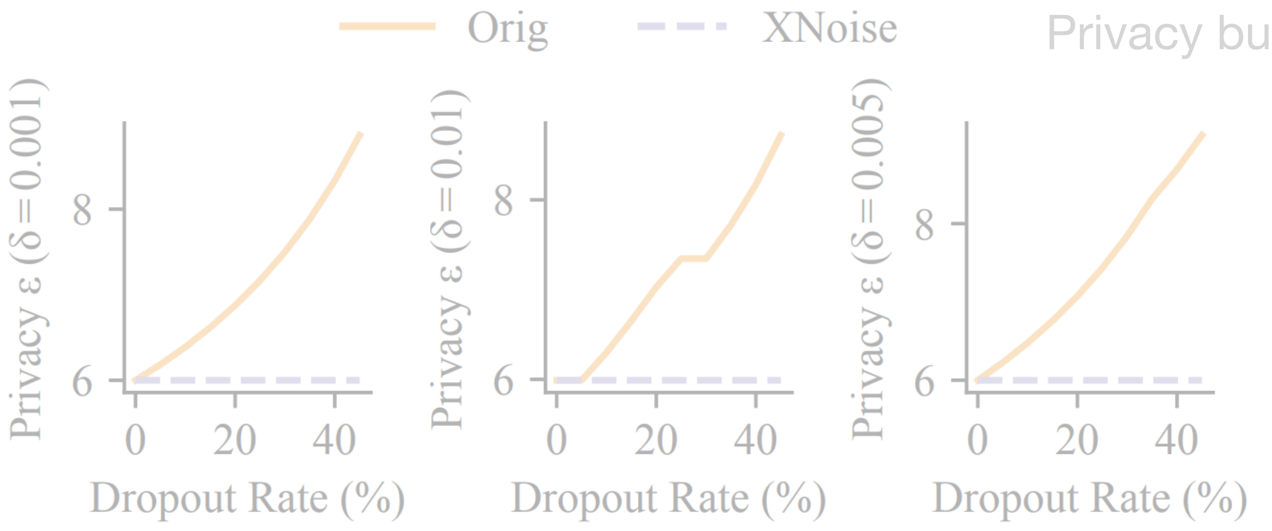
Datasets

d	0		10%		20%		30%		40%	
	Ori	XNo	Ori	XNo	Ori	XNo	Ori	XNo	Ori	XNo
F	61.3	61.4	61.4	61.4	61.2	61.4	61.2	61.2	61.4	61.5
C	66.5	66.3	66.7	66.9	66.6	65.7	64.3	65.7	63.8	64.2
R	2169	2142	2158	2179	2286	2285	2294	2317	2299	2329

Dropout-resilient noise enforcement

Effectiveness

Improves privacy



Privacy budget = 6

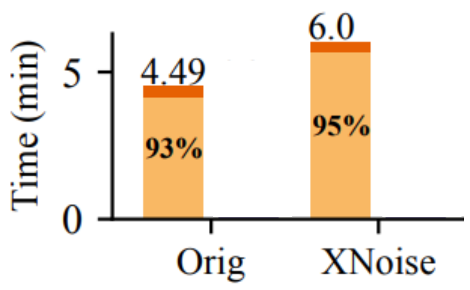
(a) FEMNIST. (b) CIFAR-10. (c) Reddit.

without sacrificing final model utility

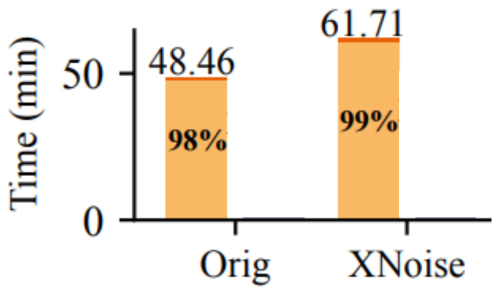
Dropout rates

Datasets	*d*	0		10%		20%		30%		40%	
		Ori	XNo	Ori	XNo	Ori	XNo	Ori	XNo	Ori	XNo
	F	61.3	61.4	61.4	61.4	61.2	61.4	61.2	61.2	61.4	61.5
	C	66.5	66.3	66.7	66.9	66.6	65.7	64.3	65.7	63.8	64.2
	R	2169	2142	2158	2179	2286	2285	2294	2317	2299	2329

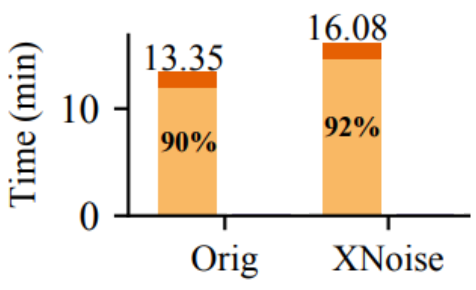
and incurs acceptable ($\leq 34\,\%$) runtime cost



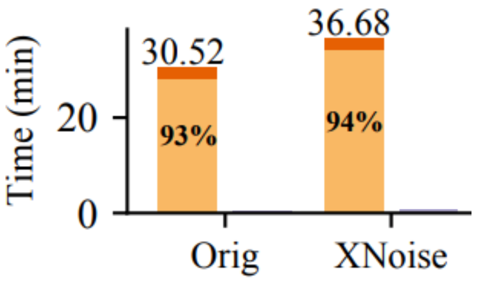
FEMNIST@CNN



FEMNIST@ResNet18



CIFAR10@ResNet18



CIFAR10@VGG19

Example: no dropout

Three practical issues in distributed DP

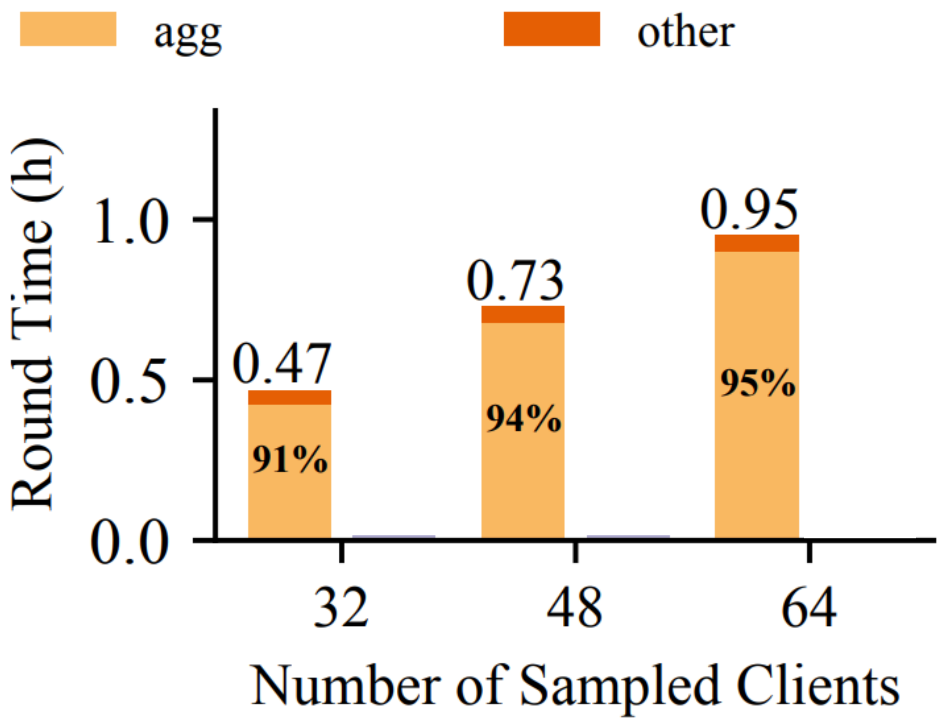
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
    - Extensive use of secret sharing and pairwise masking

Three practical issues in distributed DP

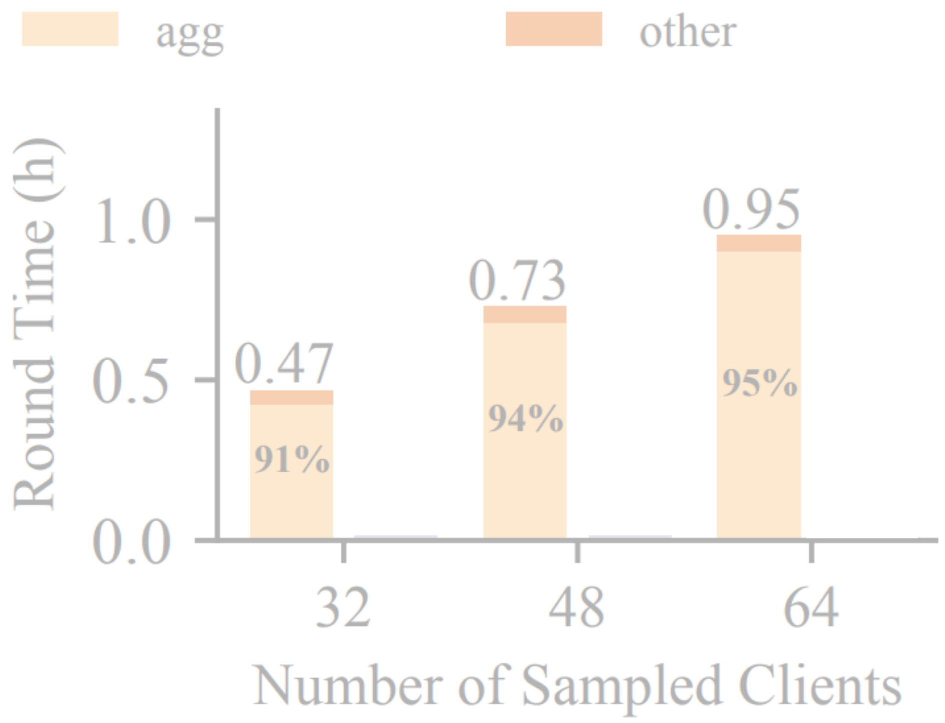
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
   - Extensive use of secret sharing and pairwise masking
   - Dominates the training time (at least 91%)



original secure aggregation: SecAgg

Three practical issues in distributed DP

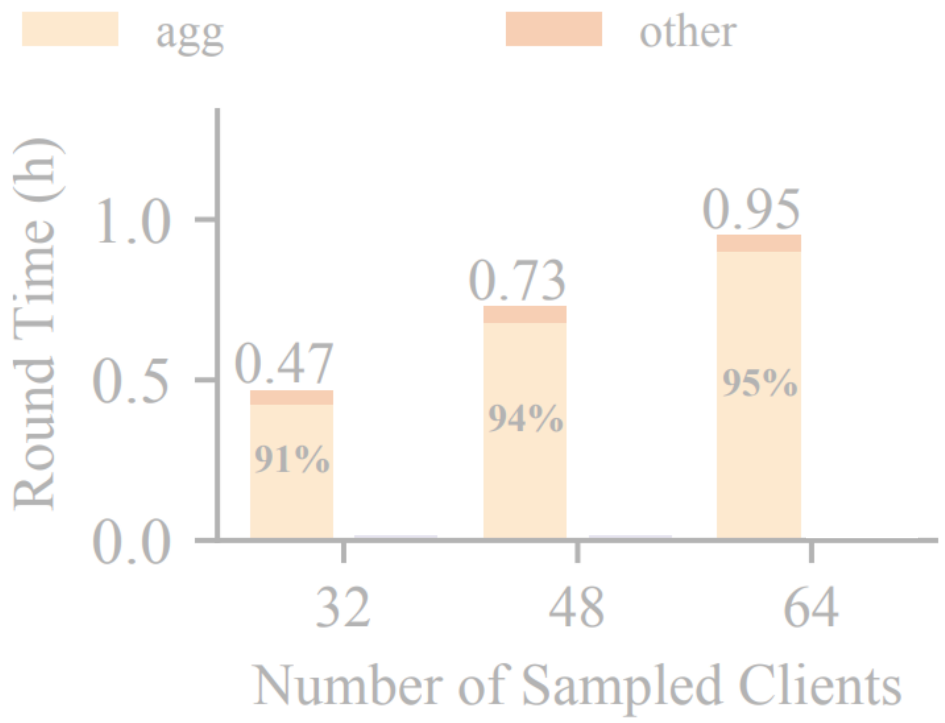
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
    - Extensive use of secret sharing and pairwise masking
    - Dominates the training time (at least 91%)
    - Follow-up solutions
        - e.g. SecAgg+: improves asymptotically



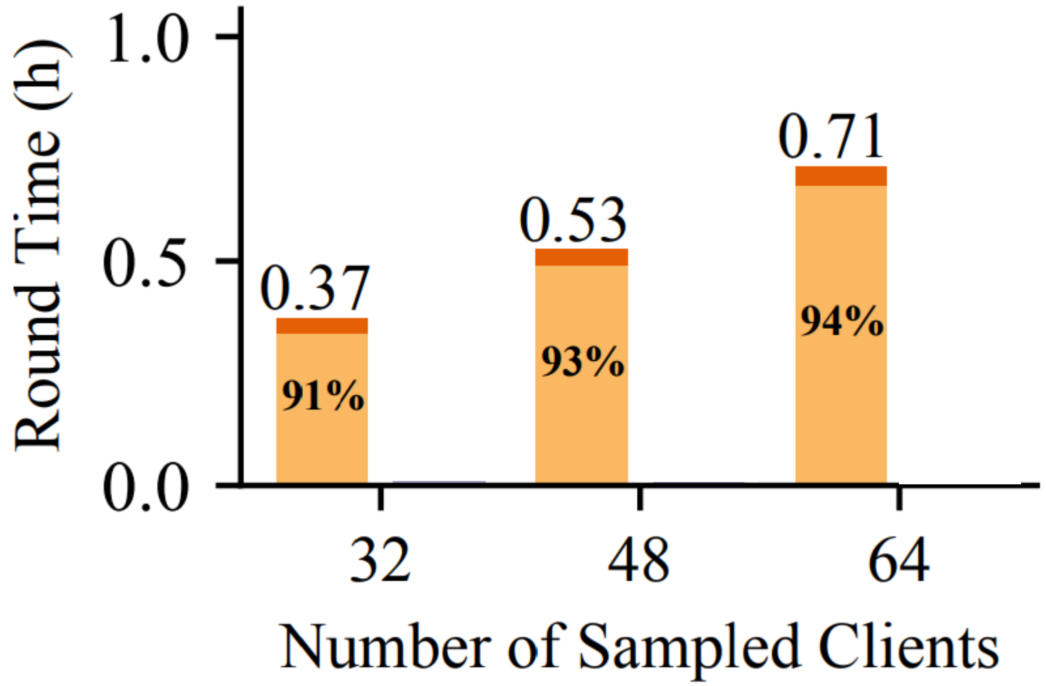
original secure aggregation: SecAgg

Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
   - Extensive use of secret sharing and pairwise masking
   - Dominates the training time (at least 91%)
   - Follow-up solutions have inefficiencies
     - e.g. SecAgg+: improves asymptotically, but not so helpful in small-scale practice[1]



original secure aggregation: SecAgg



SOTA secure aggregation: SecAgg+

[1] Towards federated learning at scale: system design, MLSys '19

Pipeline-parallel acceleration

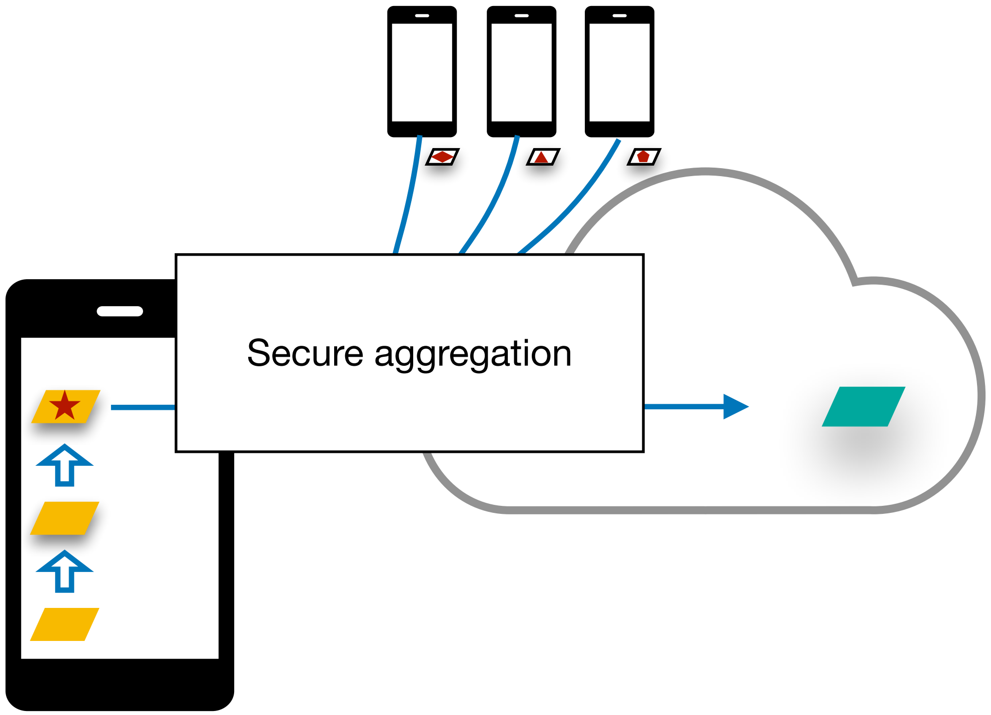
Goal: leverage the underutilized resources in the system level

Pipeline-parallel acceleration

Goal: leverage the underutilized resources in the system level

Approach:

- Step 1: Identify the types of system resources



☁ **s-comp**: *the compute resources (e.g., CPU, GPU, and memory) of the server*

📱 **c-comp**: *the compute resources of clients*

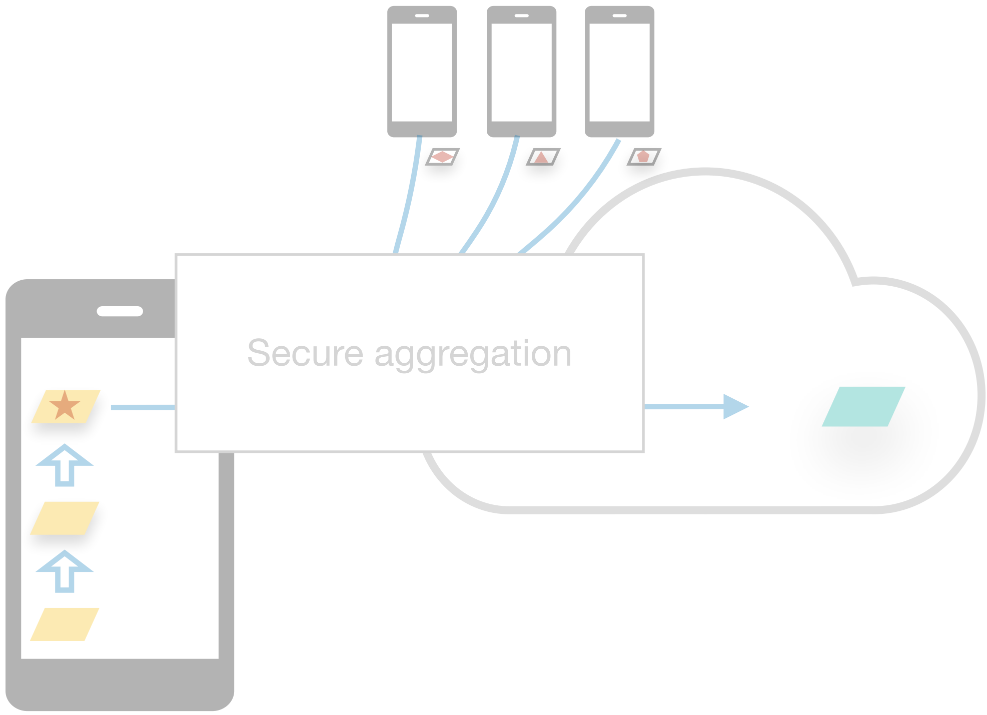
↔ **comm**: *the network resource used for server-client communication*

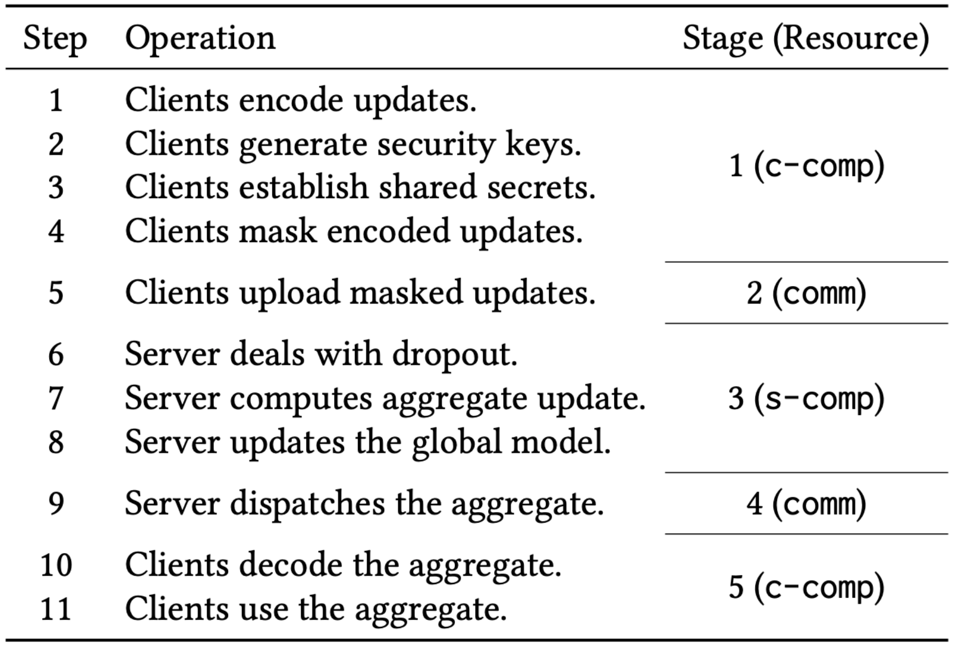
Pipeline-parallel acceleration

Goal: leverage the underutilized resources in the system level

Approach:

- Step 1: Identify the types of system resources
- Step 2: Group consecutive operations that use the same system resources





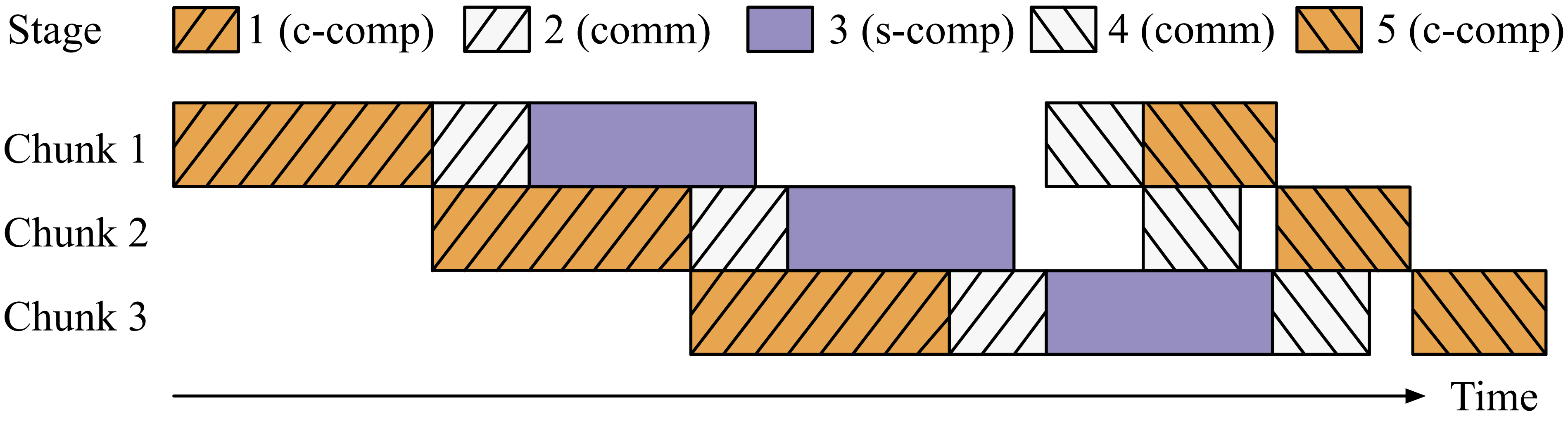
Step	Operation	Stage (Resource)
1	Clients encode updates.	1 (c-comp)
2	Clients generate security keys.	
3	Clients establish shared secrets.	
4	Clients mask encoded updates.	
5	Clients upload masked updates.	2 (comm)
6	Server deals with dropout.	3 (s-comp)
7	Server computes aggregate update.	
8	Server updates the global model.	
9	Server dispatches the aggregate.	4 (comm)
10	Clients decode the aggregate.	5 (c-comp)
11	Clients use the aggregate.	

Pipeline-parallel acceleration

Goal: leverage the underutilized resources in the system level

Approach:

- Step 1: Identify the types of system resources
- Step 2: Group consecutive operations that use the same system resources
- Step 3: Evenly partition each client's update into chunks and pipeline their processing



Pipeline-parallel acceleration

Goal: leverage the underutilized resources in the system level

Approach:

- Step 1: Identify the types of system resources
- Step 2: Group consecutive operations that use the same system resources
- Step 3: Evenly partition each client's update into chunks and pipeline their processing
  - Solve an optimization problem to determine the optimal number of chunk, m^*

$$m^* = \arg\min_{m \in N_+} f_{a,m}$$

$$s.t. \quad f_{s,c} = b_{s,c} + l_s$$

Definition of the finish time of Chunk m at Stage a

$$b_{s,c} = \max\{o_{s,c}, r_{s,c}\}$$

$$o_{s,c} = \begin{cases} 0, & \text{if } s = 0, \\ f_{s-1,c} \end{cases}$$

Intra-chunk sequential execution

$$r_{s,c} = \begin{cases} 0, & \text{if } s = 0 \text{ and } c = 0, \\ f_{q,m} \text{ or } \perp, & \text{if } s \neq 0 \text{ and } c = 0, \\ f_{s,c-1}, & \text{otherwise} \end{cases}$$

Exclusive allocation & Inter-chunk sequential execution

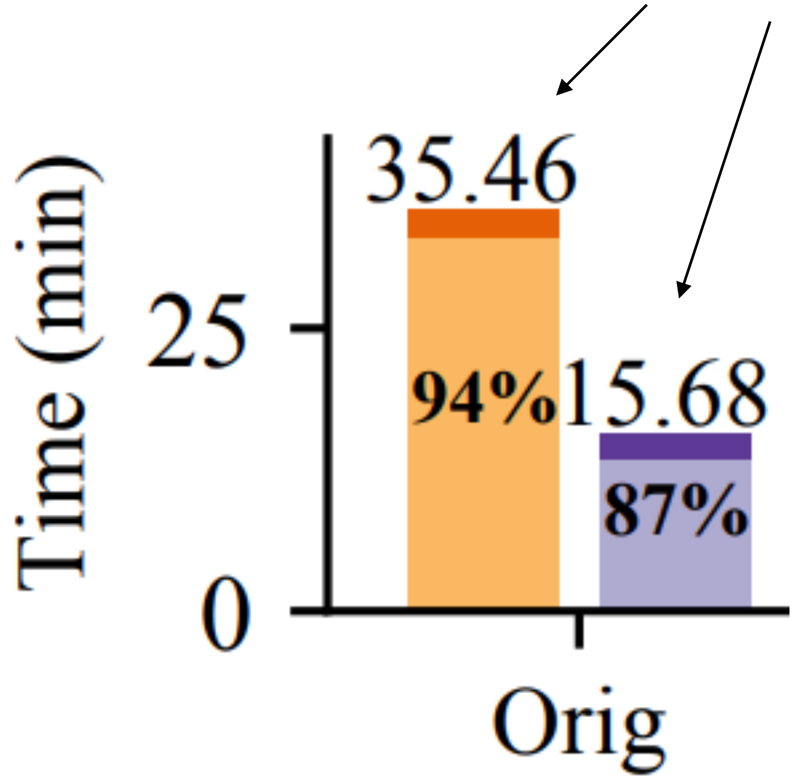
Pipeline-parallel acceleration

Effectiveness:

① A maximum speedup of 2.4×

w/ or w/o

pipelining

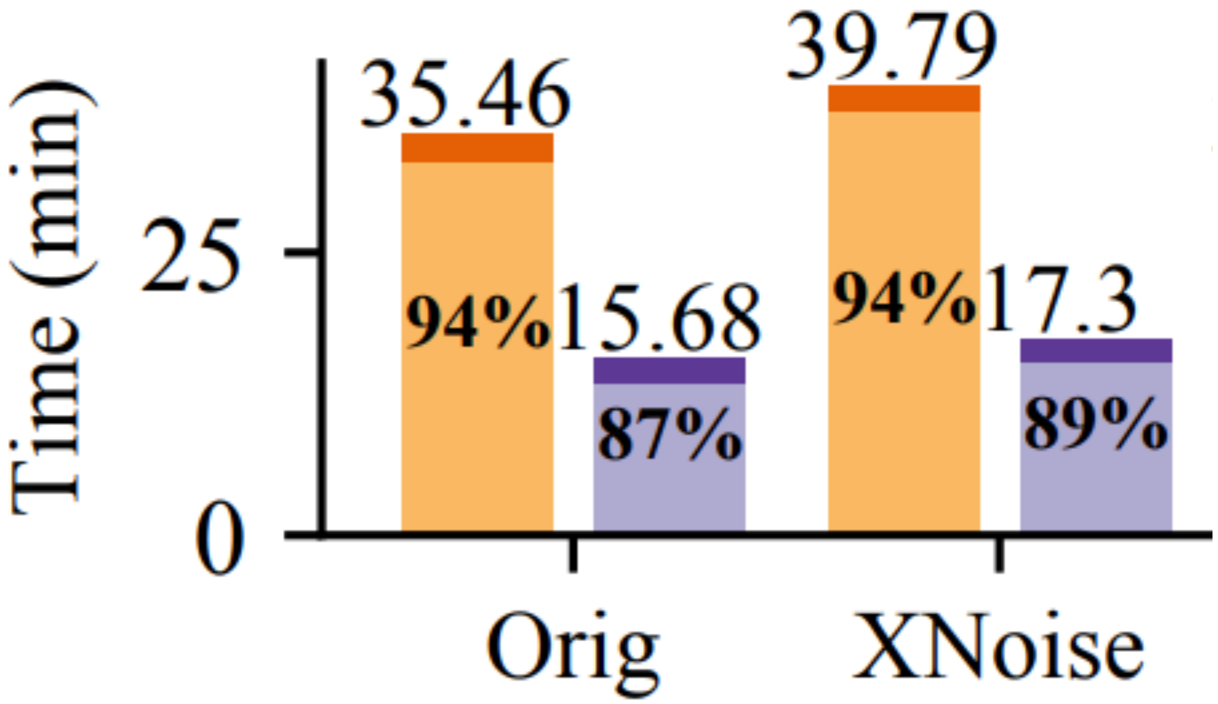


Case study: CIFAR10 @
VGG19, dropout rate = 30%

Pipeline-parallel acceleration

Effectiveness:

① A maximum speedup of 2.4×



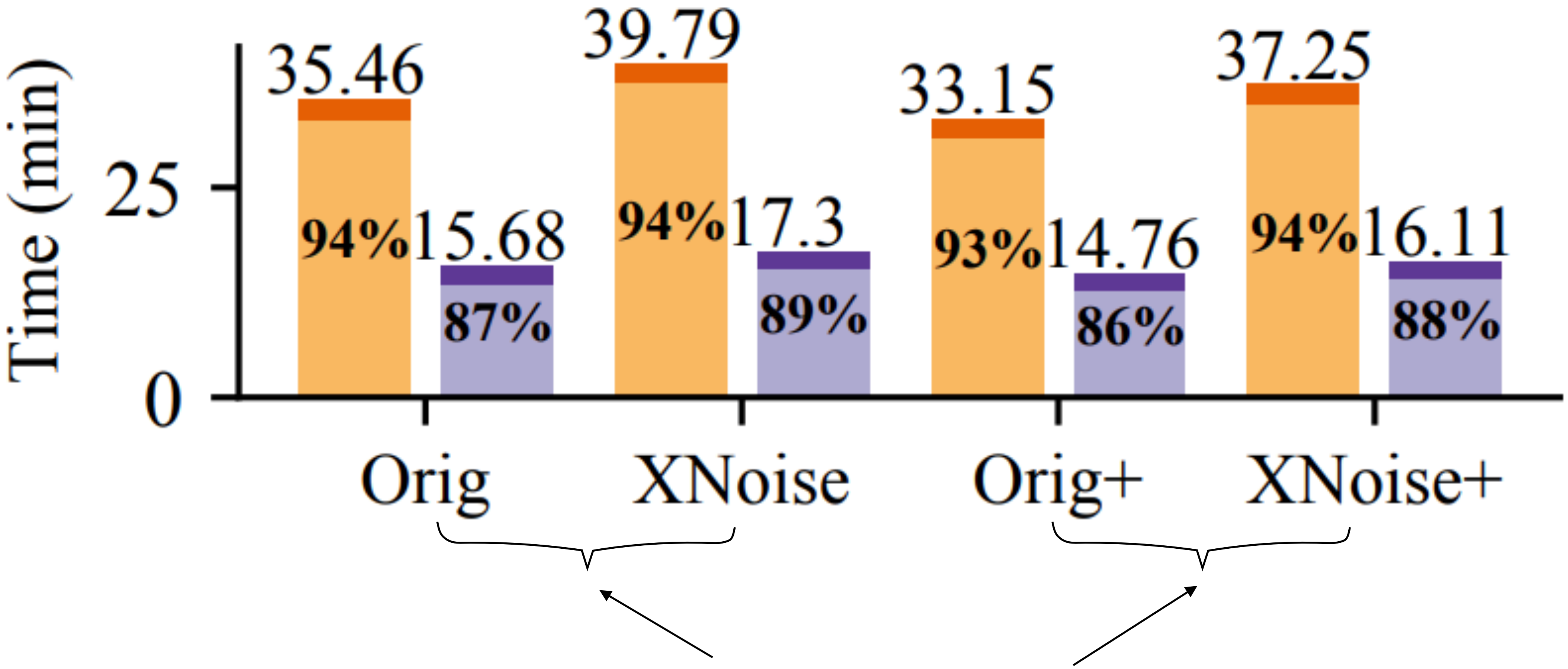
w/o or w/
our noise enforcement

Case study: CIFAR10 @
VGG19, dropout rate = 30%

Pipeline-parallel acceleration

Effectiveness:

① A maximum speedup of 2.4×



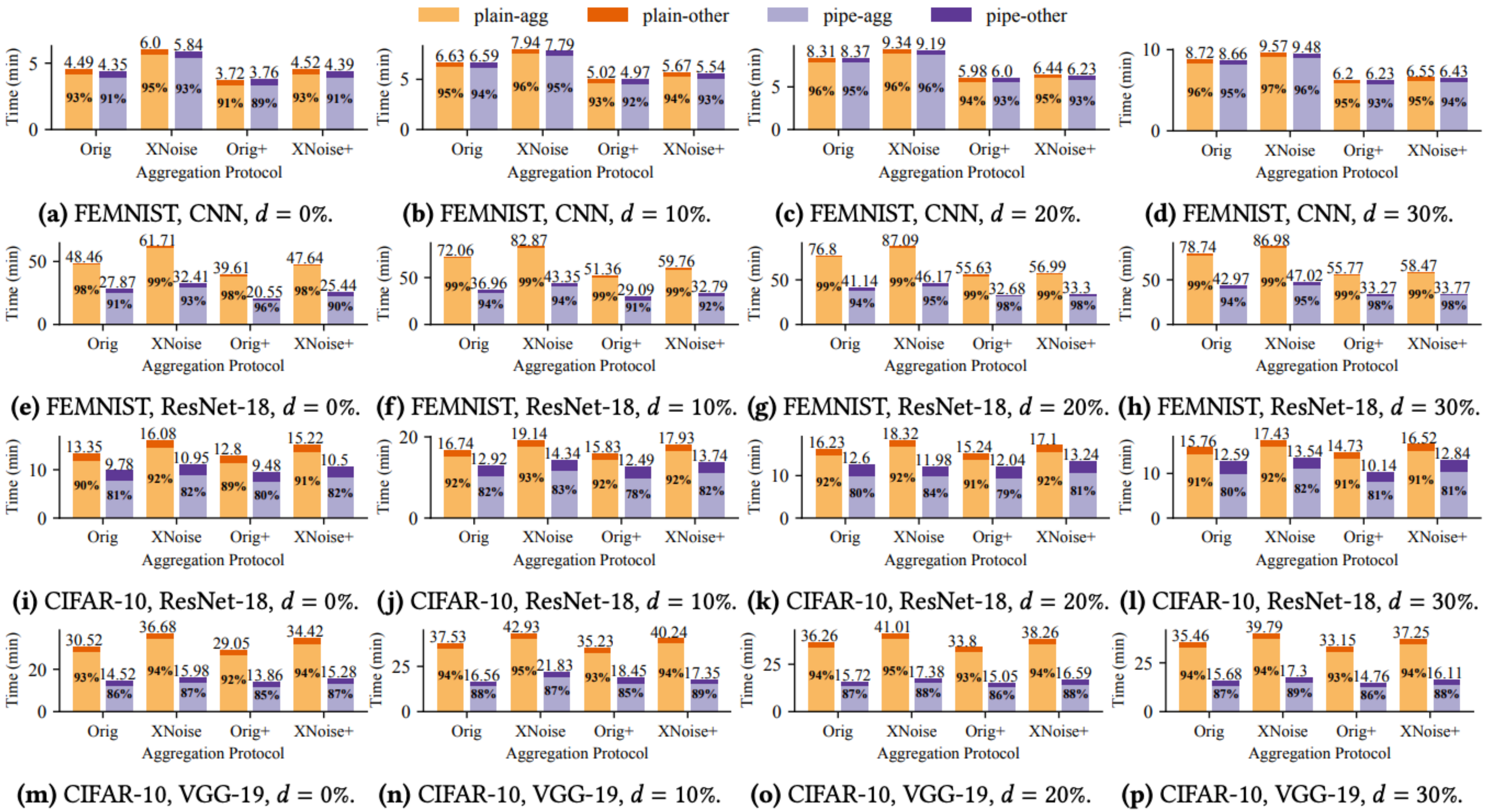
Implemented using
SecAgg or SecAgg+

Case study: CIFAR10 @
VGG19, dropout rate = 30%

Pipeline-parallel acceleration

Effectiveness:

① A maximum speedup of 2.4×



(a) FEMNIST, CNN, $d = 0\%$. (b) FEMNIST, CNN, $d = 10\%$. (c) FEMNIST, CNN, $d = 20\%$. (d) FEMNIST, CNN, $d = 30\%$.

(e) FEMNIST, ResNet-18, $d = 0\%$. (f) FEMNIST, ResNet-18, $d = 10\%$. (g) FEMNIST, ResNet-18, $d = 20\%$. (h) FEMNIST, ResNet-18, $d = 30\%$.

(i) CIFAR-10, ResNet-18, $d = 0\%$. (j) CIFAR-10, ResNet-18, $d = 10\%$. (k) CIFAR-10, ResNet-18, $d = 20\%$. (l) CIFAR-10, ResNet-18, $d = 30\%$.

(m) CIFAR-10, VGG-19, $d = 0\%$. (n) CIFAR-10, VGG-19, $d = 10\%$. (o) CIFAR-10, VGG-19, $d = 20\%$. (p) CIFAR-10, VGG-19, $d = 30\%$.

1M

11M

11M

20M

② Larger models gain more

100

16

③ It scales with number of participants

④ The gains are consistent across different dropout rates

Three practical issues in distributed DP

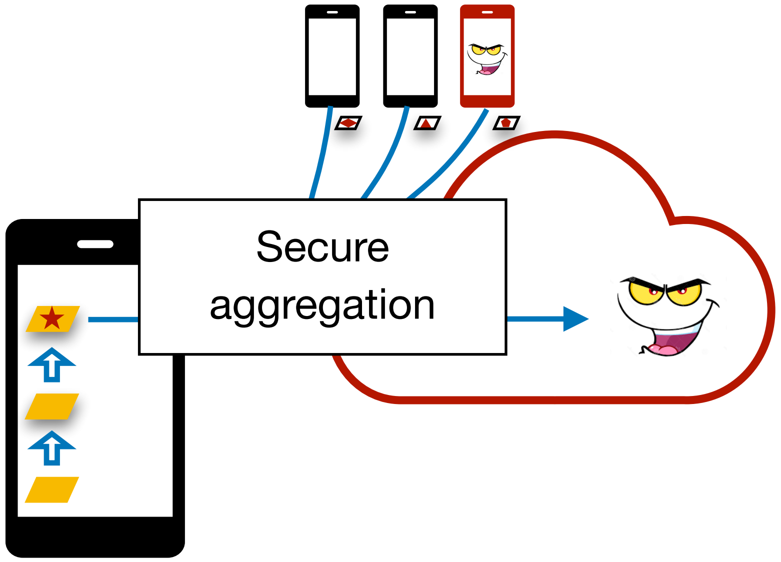
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive nature of secure aggregation

✉ Distributed DP can be made more practical,
by enforcing target privacy in the presence of client dropout
and optimizing execution efficiency.

🏆 EuroSys ‘24

Three practical issues in distributed DP

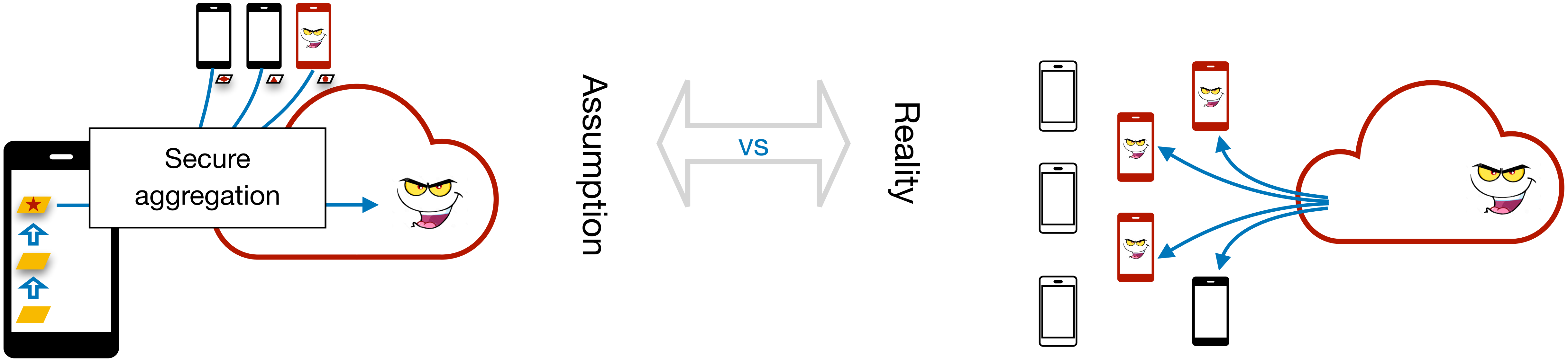
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants



Assumption

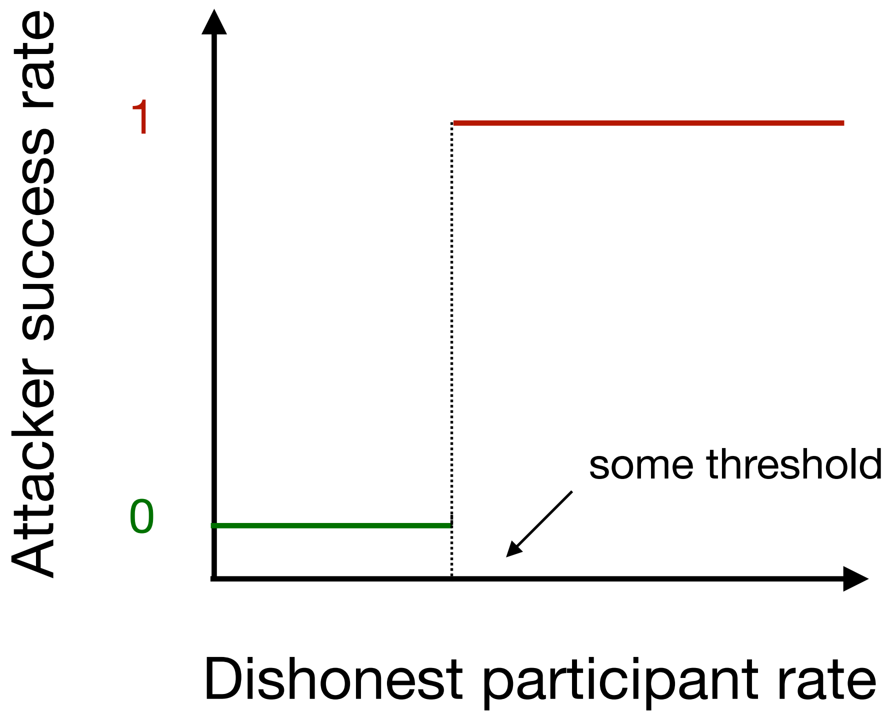
Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants
   - Adversarial server can game participant selection



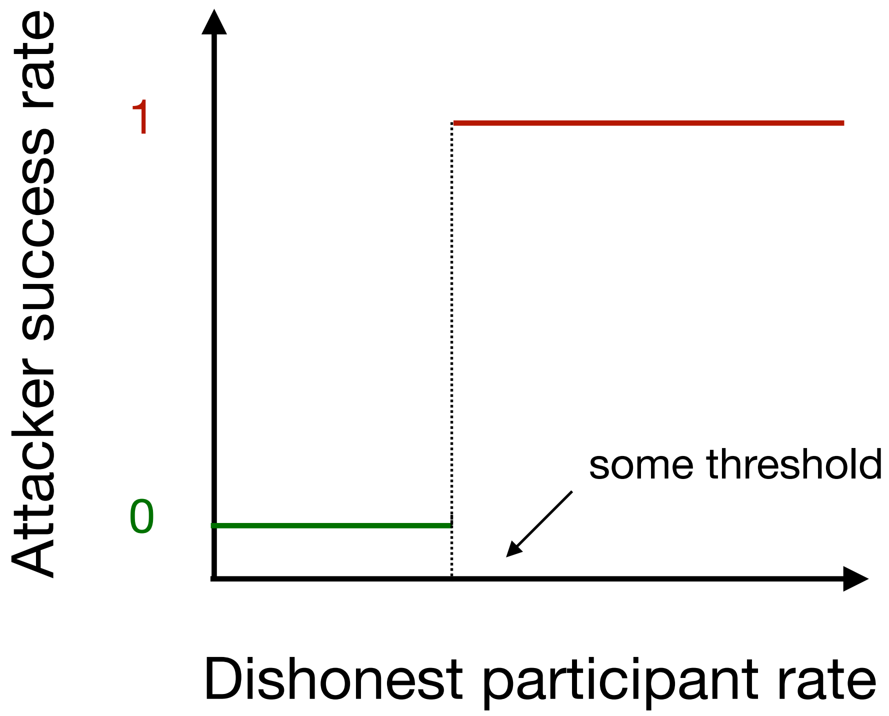
Three practical issues in distributed DP

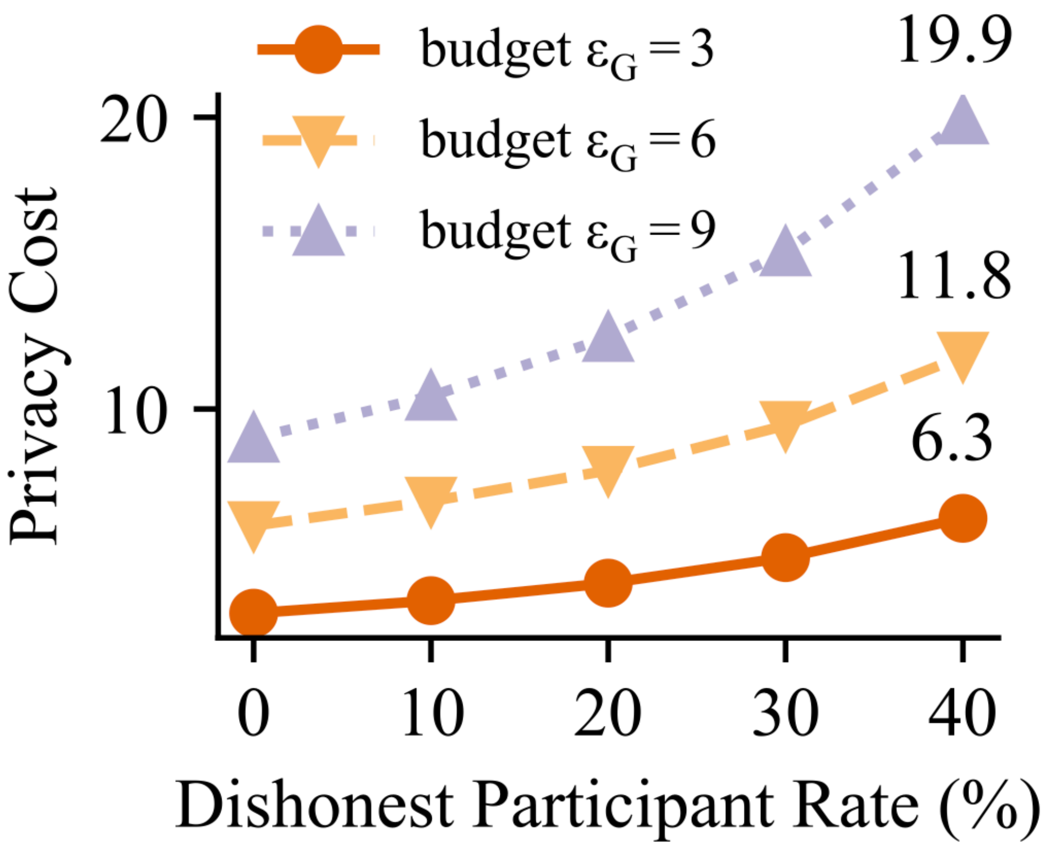
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants
   - Adversarial server can game participant selection
   - Secure aggregation breaks



Three practical issues in distributed DP

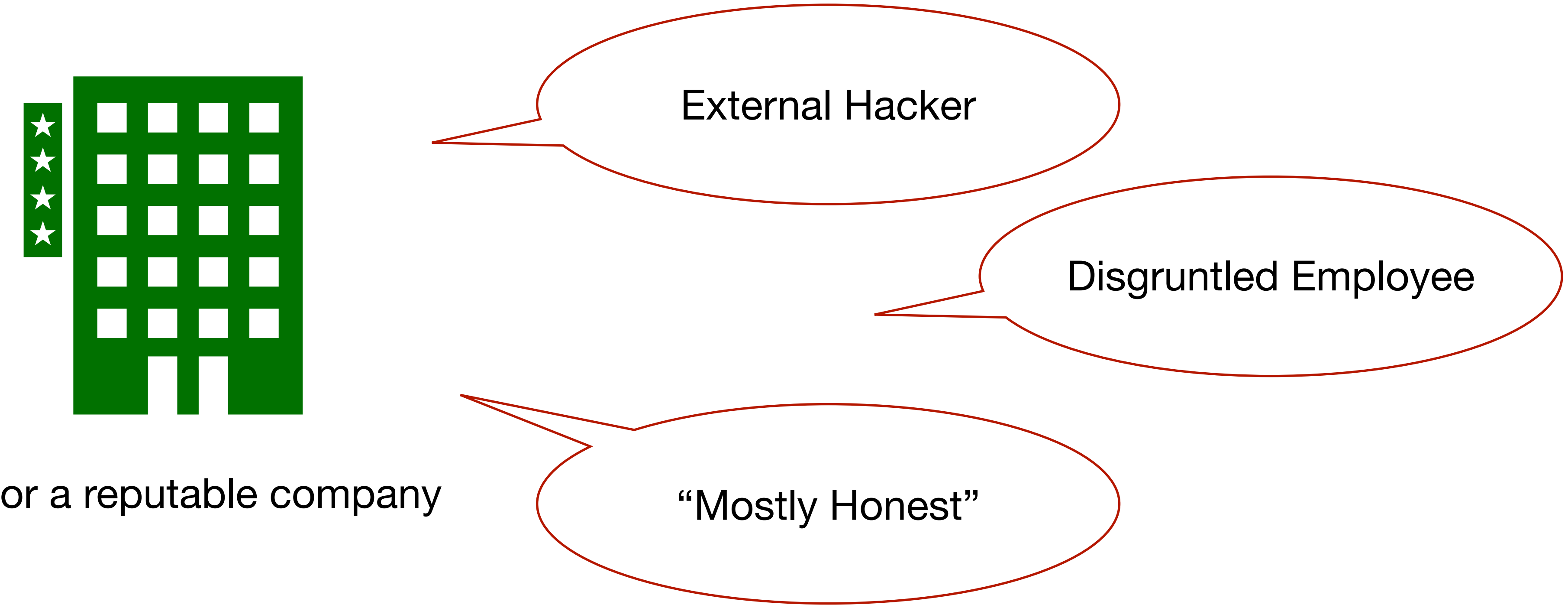
1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants
   - Adversarial server can game participant selection
   - Secure aggregation breaks; distributed DP degrades





Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants
    - Adversarial server can game participant selection
    - Secure aggregation breaks; distributed DP degrades
    - The problem has been overlooked:



External Hacker

Disgruntled Employee

"Mostly Honest"

Even for a reputable company

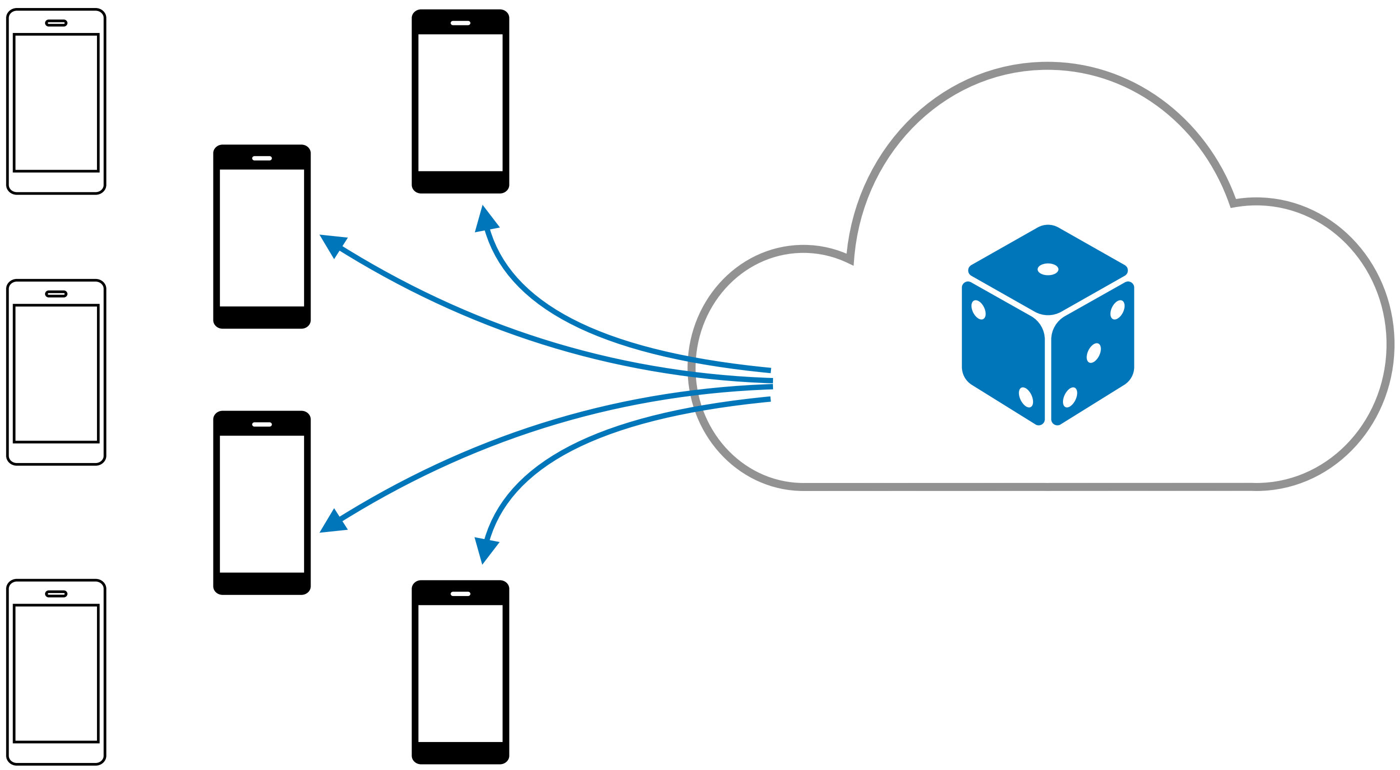
Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

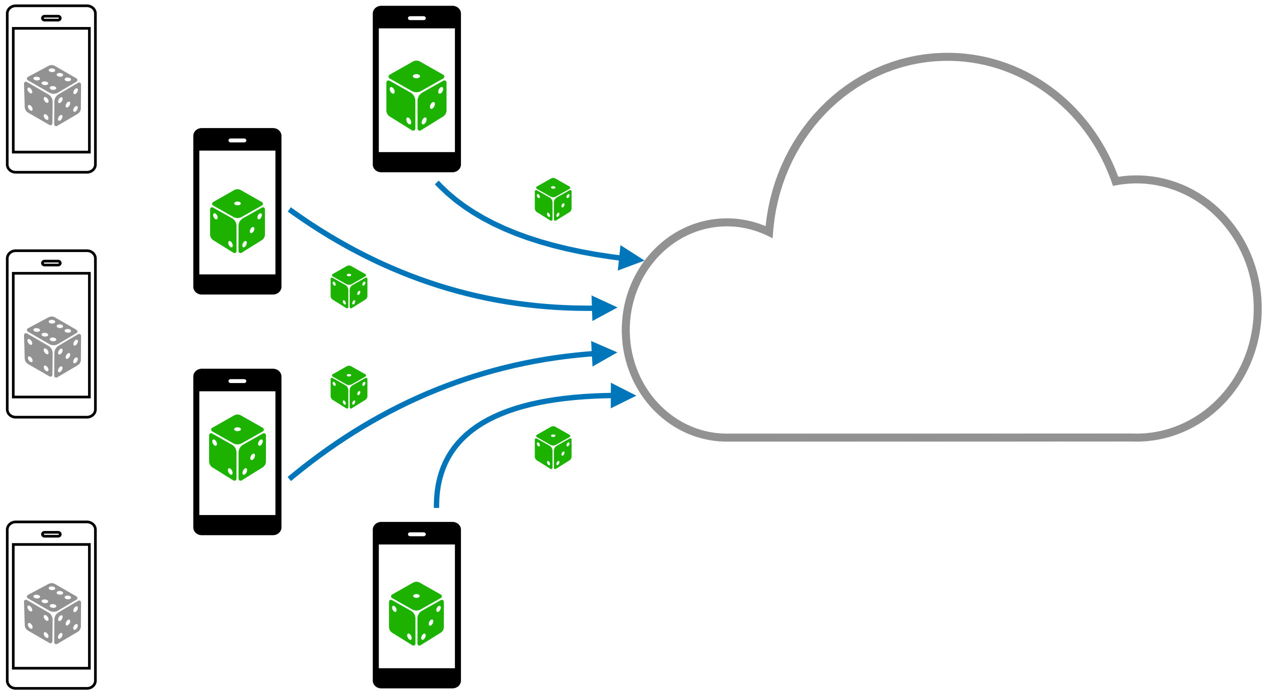


Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
  - Each client i in the population
  - Join if $r_i \in [0,R) < pR$ for some $p \in (0,1)$

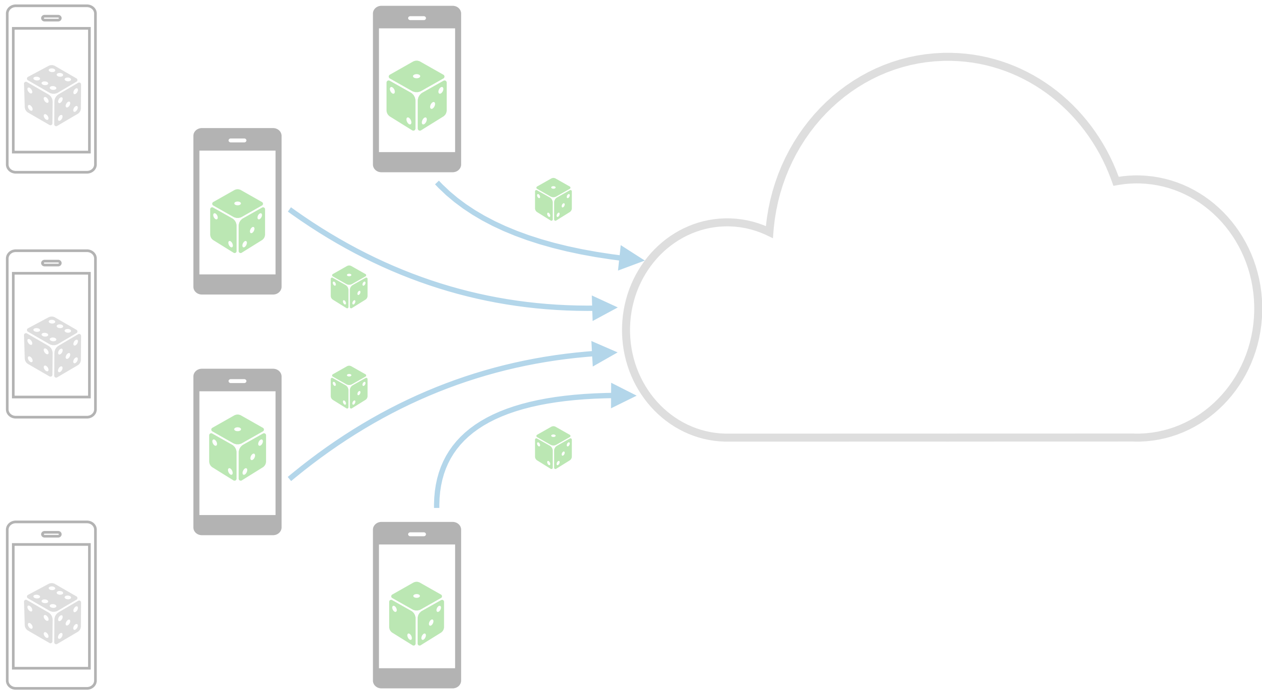


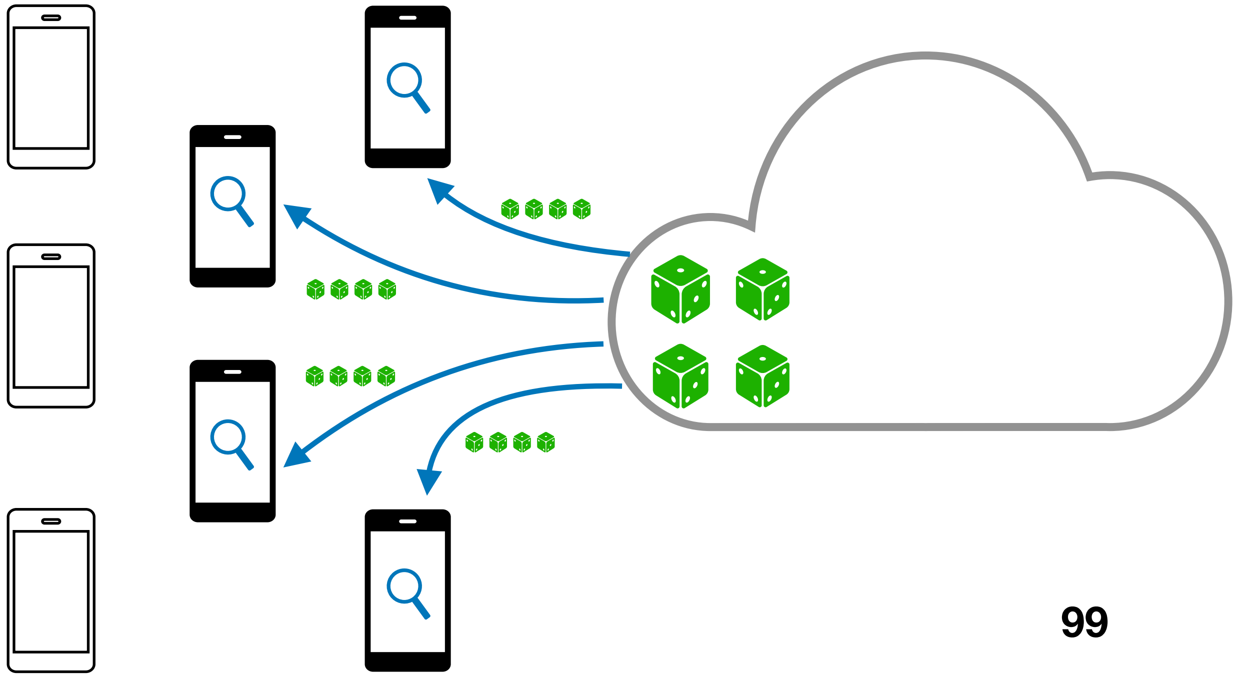
Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
    - Each client i claiming to join
    - Proceed only if $r_j < pR$ for $\forall j \neq i$



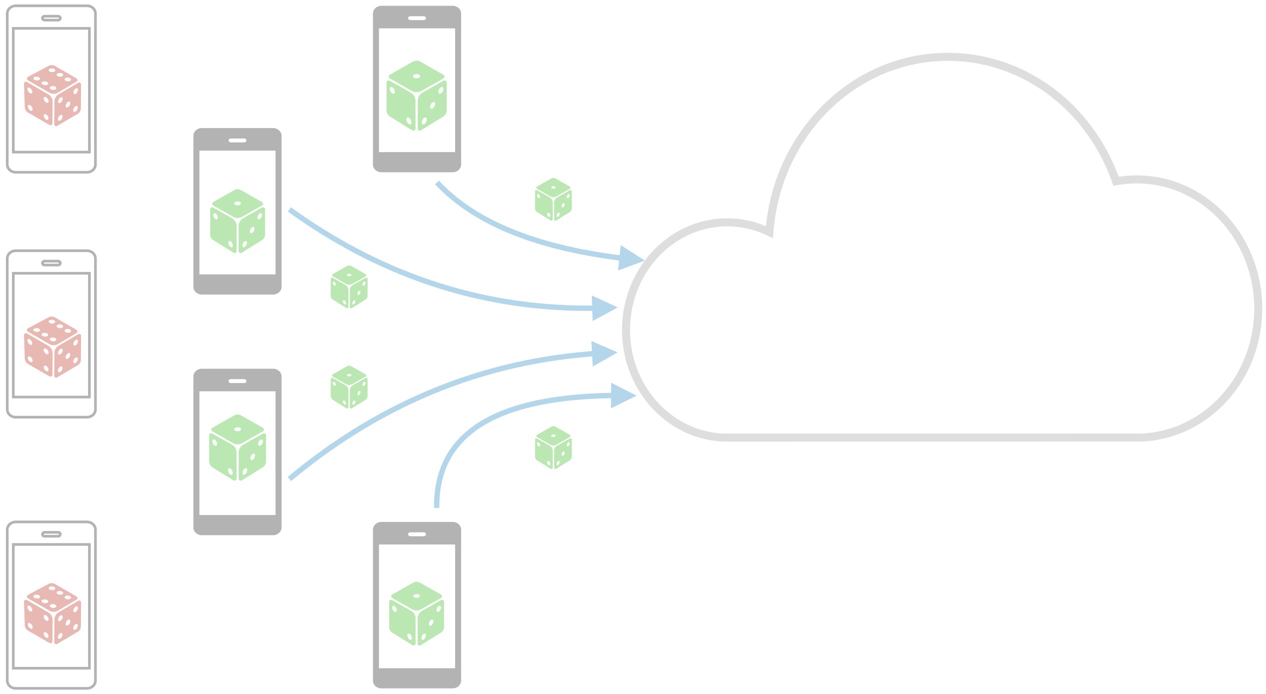


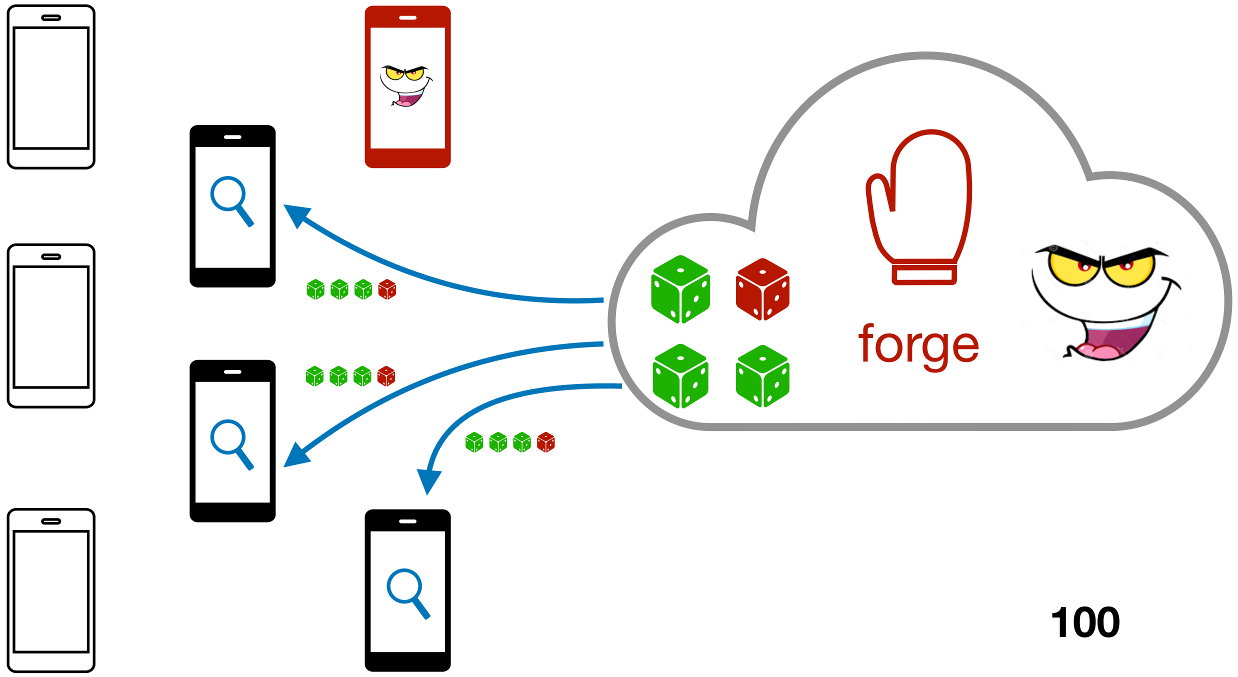
Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
  - Each client i claiming to join
  - Proceed only if $r_j < pR$ for $\forall j \neq i$





Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)
  - Assume each client i has a key pair (sk_i, pk_i) with integrity guaranteed by a PKI
  - For each $j \neq i$, client i also verifies that VRF.ver$(pk_j, r, \beta_j, \pi_j) = 1$
  - The test passes only if $\beta_j, \pi_j =$ VRF.eval(sk_j, r)

secret key, bound to j

Round index, public

Neither can be manipulated!

Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

Secure informed selection

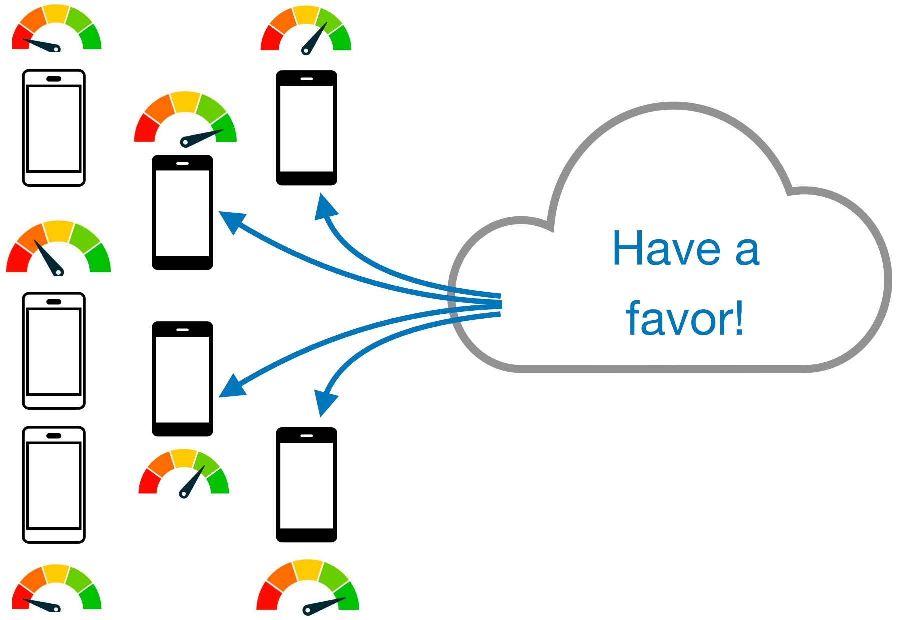
Self-sampling with verifiable randomness

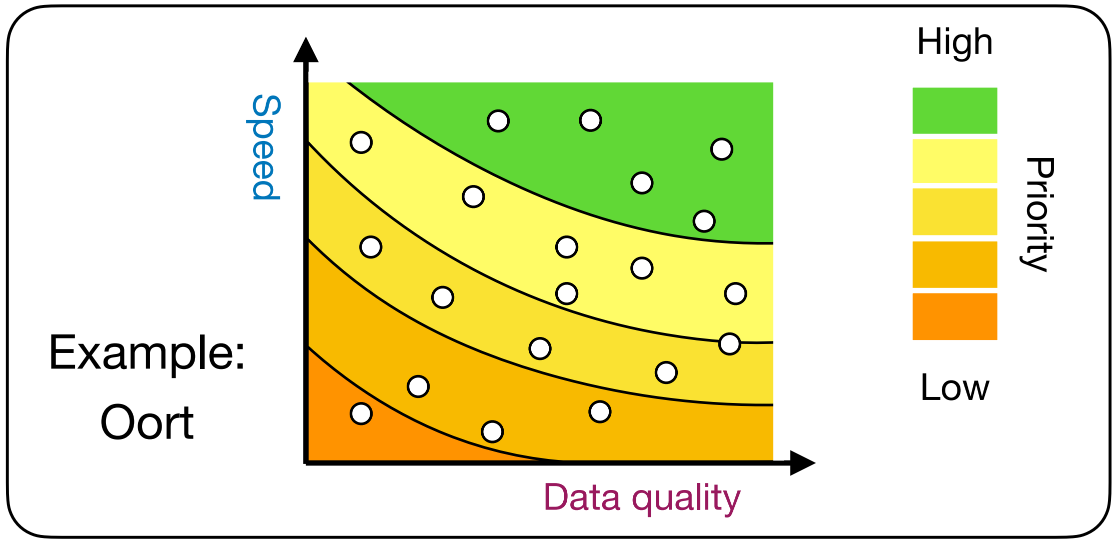
Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

Secure informed selection





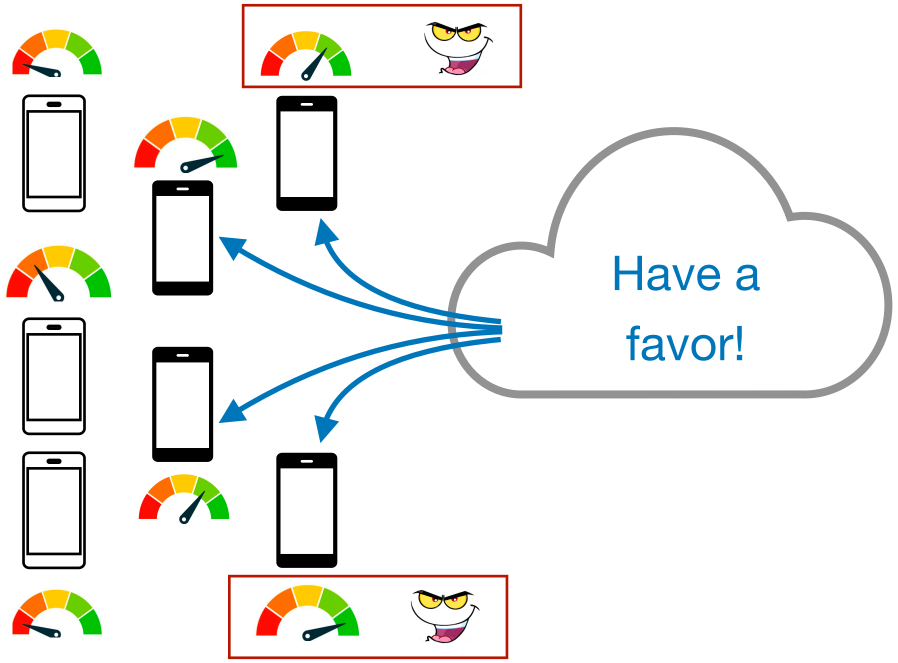
Self-sampling with verifiable randomness

Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

Secure informed selection



Gaming

- Direct: refer to fake metrics
- Indirect: optimize the referred metrics

Self-sampling with verifiable randomness

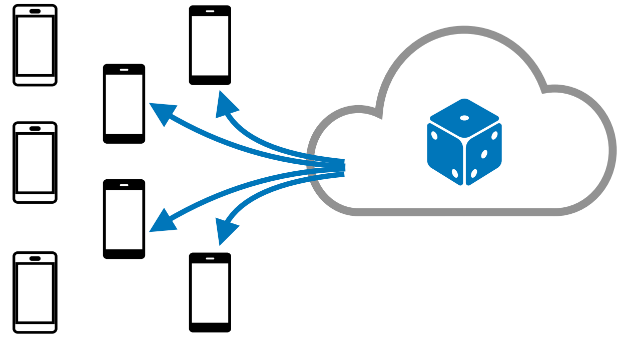
Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

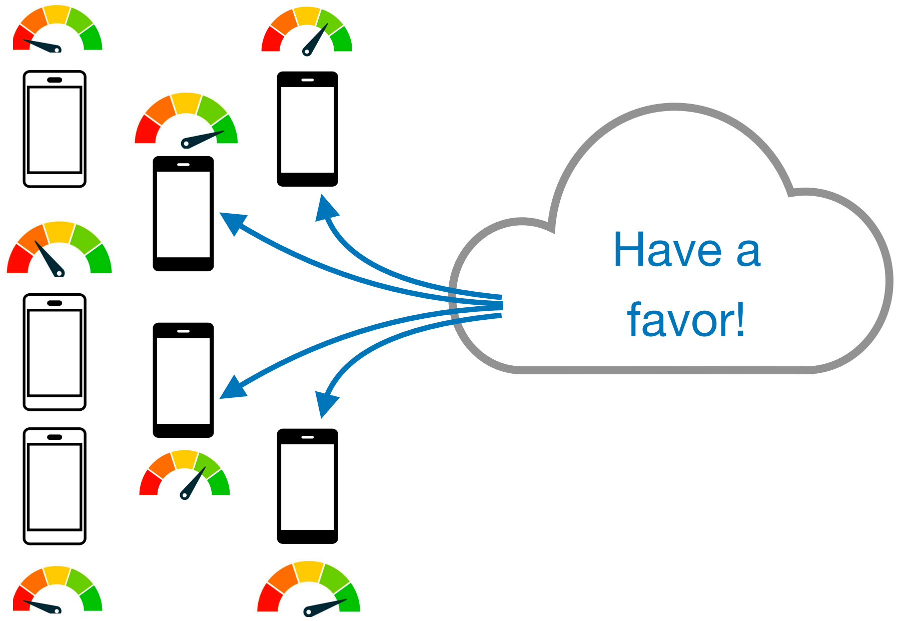
Secure informed selection

- Prevent gaming: verifiable randomness has to be introduced to the last mile



Secure random selection

≈



Self-sampling with verifiable randomness

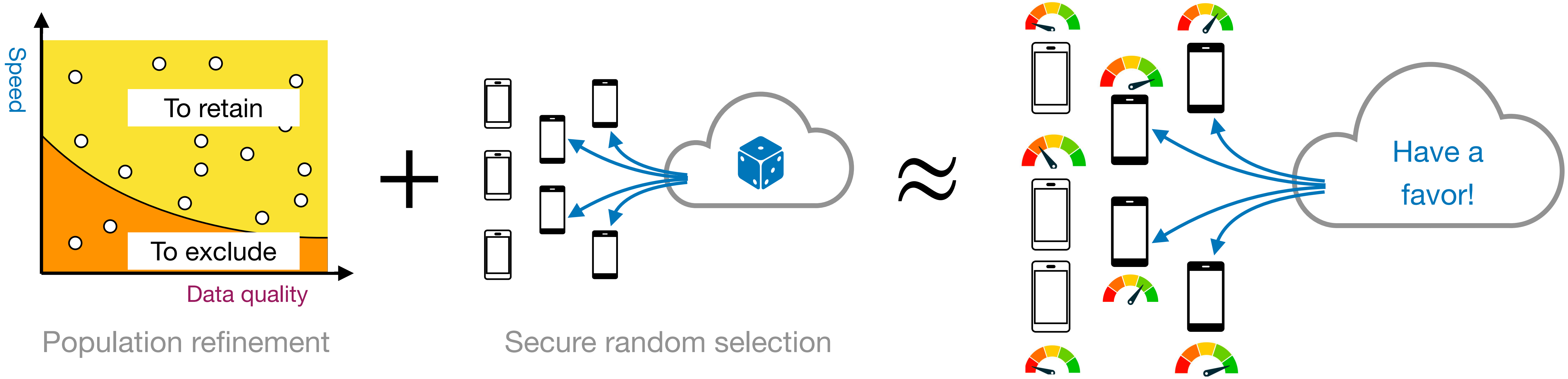
Goal: to know whether the server manipulates the selection

Secure random selection

- Self-sampling
- Mutual verification
- Prevent forging: verifiable random functions (VRFs)

Secure informed selection

- Prevent gaming: verifiable randomness has to be introduced to the last mile
- Achieve the expected effect of selection: the server refine the population in advance



Population refinement

Secure random selection

Lotto: Self-sampling with verifiable randomness

Effectiveness

Random selection

① Provably **aligns** the fractions of compromised participants **to** the base rate of dishonest clients in the population

Lotto: Self-sampling with verifiable randomness

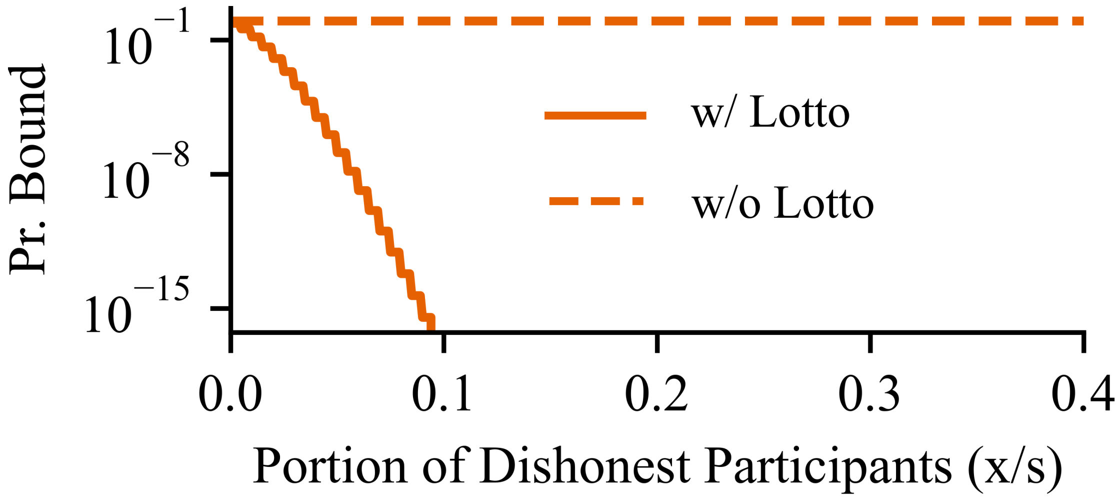
Effectiveness

Random selection

① Provably **aligns** the fractions of compromised participants **to** the base rate of dishonest clients in the population

Assumption:

- Population size n = 200k
- 0.1% dishonest clients in the population



10% dishonest clients in the participant

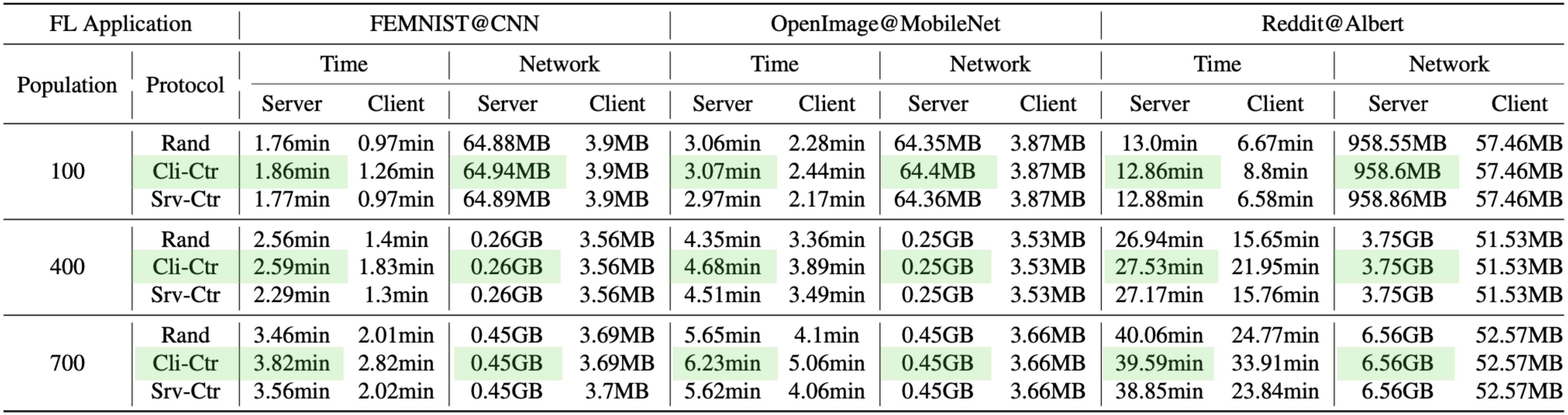
Lotto: Self-sampling with verifiable randomness

Effectiveness

Random selection

① Provably **aligns** the fractions of compromised participants **to** the base rate of dishonest clients in the population

② with acceptable runtime cost ($\leq 10\,\%$) and negligible network overhead ($\leq 1\,\%$)



FL Application		FEMNIST@CNN				OpenImage@MobileNet				Reddit@Albert			
Population	Protocol	Time		Network		Time		Network		Time		Network	
		Server	Client	Server	Client	Server	Client	Server	Client	Server	Client	Server	Client
100	Rand	1.76min	0.97min	64.88MB	3.9MB	3.06min	2.28min	64.35MB	3.87MB	13.0min	6.67min	958.55MB	57.46MB
	Cli-Ctr	1.86min	1.26min	64.94MB	3.9MB	3.07min	2.44min	64.4MB	3.87MB	12.86min	8.8min	958.6MB	57.46MB
	Srv-Ctr	1.77min	0.97min	64.89MB	3.9MB	2.97min	2.17min	64.36MB	3.87MB	12.88min	6.58min	958.86MB	57.46MB
400	Rand	2.56min	1.4min	0.26GB	3.56MB	4.35min	3.36min	0.25GB	3.53MB	26.94min	15.65min	3.75GB	51.53MB
	Cli-Ctr	2.59min	1.83min	0.26GB	3.56MB	4.68min	3.89min	0.25GB	3.53MB	27.53min	21.95min	3.75GB	51.53MB
	Srv-Ctr	2.29min	1.3min	0.26GB	3.56MB	4.51min	3.49min	0.25GB	3.53MB	27.17min	15.76min	3.75GB	51.53MB
700	Rand	3.46min	2.01min	0.45GB	3.69MB	5.65min	4.1min	0.45GB	3.66MB	40.06min	24.77min	6.56GB	52.57MB
	Cli-Ctr	3.82min	2.82min	0.45GB	3.69MB	6.23min	5.06min	0.45GB	3.66MB	39.59min	33.91min	6.56GB	52.57MB
	Srv-Ctr	3.56min	2.02min	0.45GB	3.7MB	5.62min	4.06min	0.45GB	3.66MB	38.85min	23.84min	6.56GB	52.57MB

Lotto: Self-sampling with verifiable randomness

Effectiveness

Random selection

① Provably **aligns** the fractions of compromised participants **to** the base rate of dishonest clients in the population

② with acceptable runtime cost ($\leq 10\,\%$) and negligible network overhead ($\leq 1\,\%$)

Informed selection

① Security, overhead: similar

② Effectiveness of approximation: achieve comparable time-to-acc?

Lotto: Self-sampling with verifiable randomness

Effectiveness

Random selection

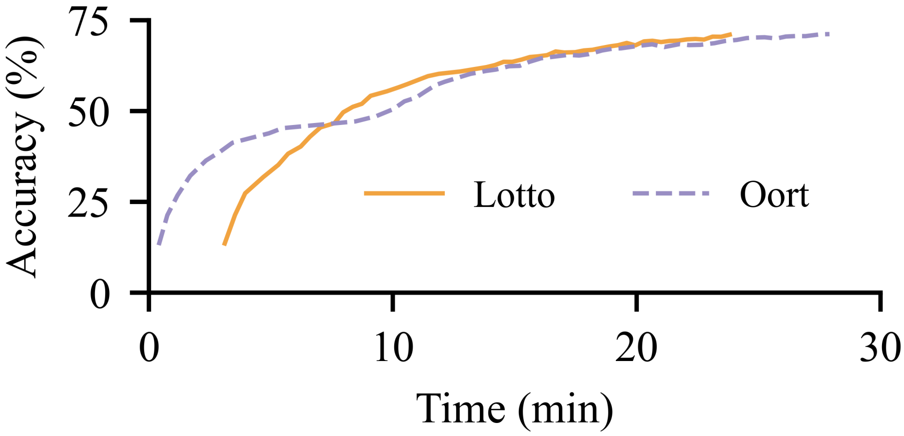
① Provably **aligns** the fractions of compromised participants **to** the base rate of dishonest clients in the population

② with acceptable runtime cost ($\leq 10\,\%$) and negligible network overhead ($\leq 1\,\%$)

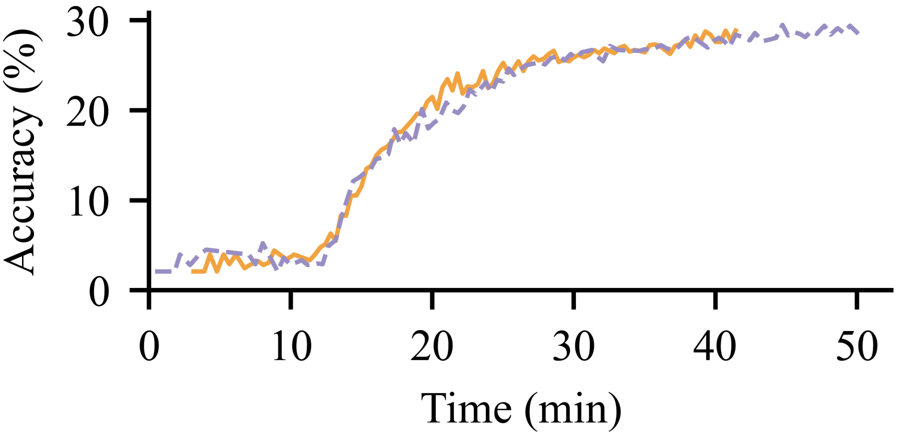
Informed selection

① Security, overhead: similar

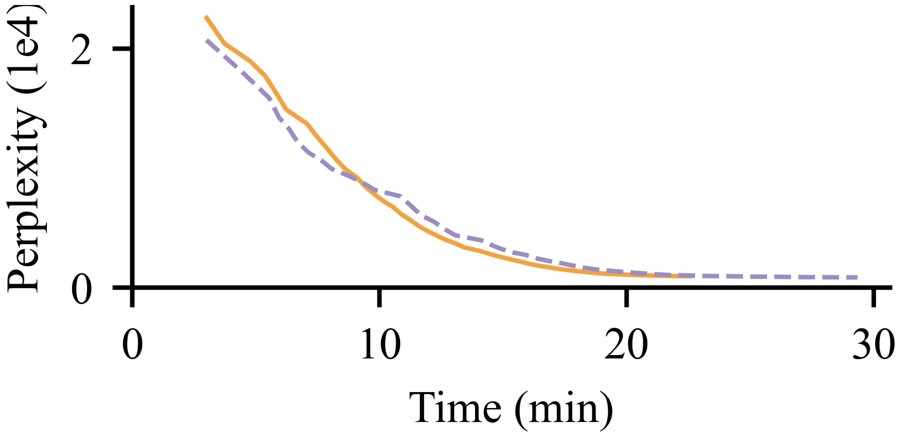
② Effectiveness of approximation: achieve comparable time-to-acc!



FEMNIST@CNN



OpenImage@MobileNet



Reddit@Albert

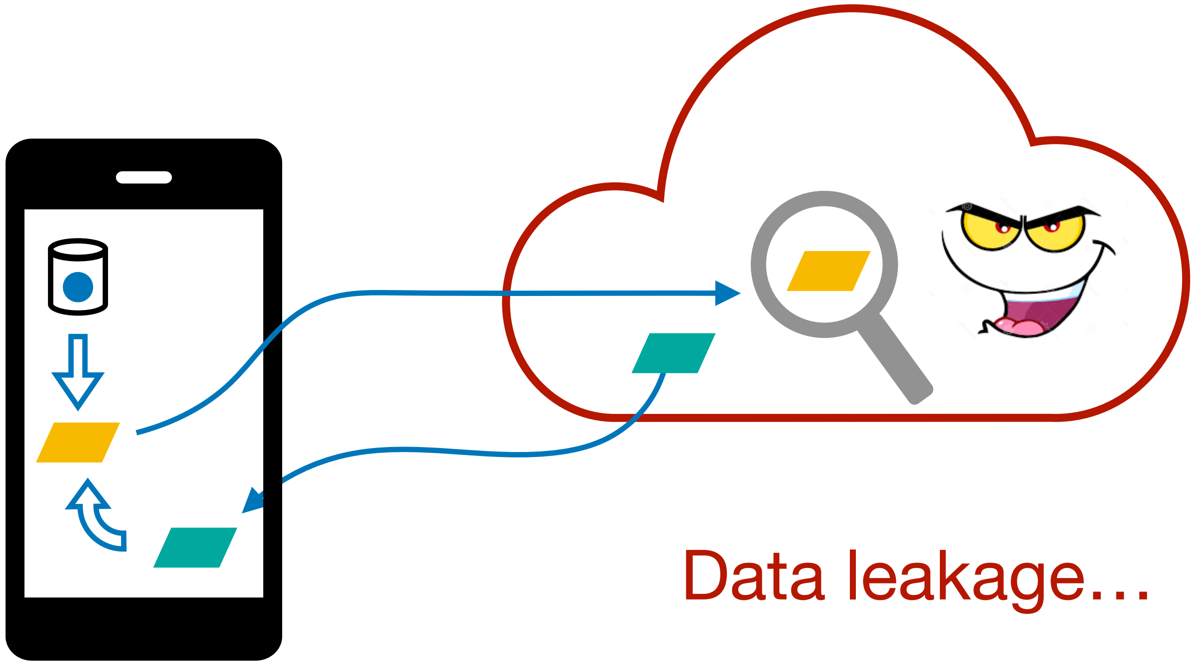
Three practical issues in distributed DP

1. **Privacy** Issue: caused by client dropout
2. **Performance** Issue: expensive use of secure aggregation
3. **Security** Issue: assume honest majority among participants

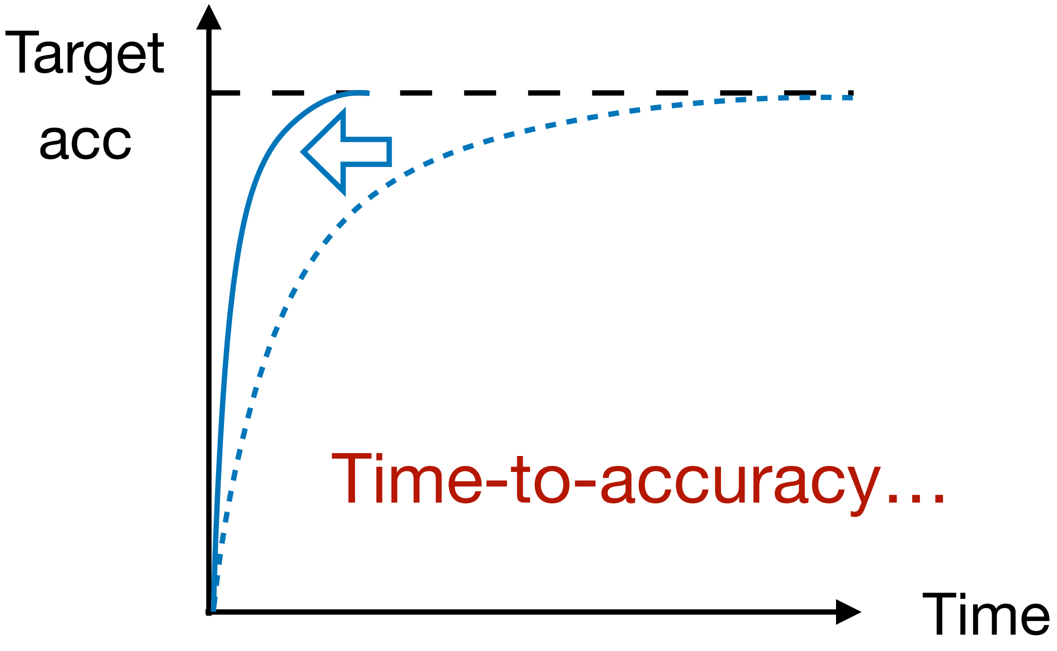
Distributed DP can be made more secure, by preventing the adversary from manipulating the participant selection process with verifiable randomness.

Security '24

My Work: build private and efficient cross-device FL



Data leakage…



Efficiency-Only

Efficient asynchronous training (SoCC ’22)

Dropout-resilient & pipeline-accelerated distributed differential privacy (EuroSys ’24)

Secure participant selection (Security ’24)

Privacy-First

Future Work (3)

Future Work (1/3)

1. **Mitigating Stragglers atop Distributed DP**
   - Existing async FL is incompatible with distributed DP
   - Straggler problems remain when distributed DP is employed
   - Existing explorations fall short in applicability/model utility

Future Work (2/3)

1. **Privacy Enhancement of Asynchronous Training**
2. **Extension of Federated Unlearning to the Participant Side**
   - Clients have the right to eliminate the impact of their data on the trained model
   - Intermediate results (e.g. aggregated updates) are also sensitive and made public
   - Existing research has overlooked this issue

Future Work (3/3)

1. Privacy Enhancement of Asynchronous Training
2. Extension of Federated Unlearning to the Participant Side
3. **Harmonizing Efficiency, Privacy and Robustness in Single-Server Scenarios**
    - The trained model is open to data poisoning and model poisoning
    - Identifying malformed local updates contradicts with the spirits of privacy protection
    - Existing remedies rely on two-server settings, which falls short in practicality

List of Publications

1. ☆ Lotto: Secure Participant Selection against Adversarial Servers in Federated Learning. ***[USENIX Security 2024]***
   - Zhifeng Jiang, Peng Ye, Shiqi He, Wei Wang, Ruichuan Chen, Bo Li
2. ☆ Dordis: Efficient Federated Learning with Dropout-Resilient Differential Privacy. ***[ACM EuroSys 2024]***
   - Zhifeng Jiang, Wei Wang, Ruichuan Chen
3. ☆ Pisces: Efficient Federated Learning via Guided Asynchronous Training. ***[ACM SoCC 2022]***
   - Zhifeng Jiang, Wei Wang, Baochun Li, Bo Li
4. Towards Efficient Synchronous Federated Training: A Survey on System Optimization Strategies. ***[IEEE Trans. Big Data 2022]***
   - Zhifeng Jiang, Wei Wang, Bo Li, Qiang Yang
5. Gillis: Serving Large Neural Networks in Serverless Functions with Automatic Model Partitioning. ***[ICDCS 2021]***
   - Minchen Yu, Zhifeng Jiang, Hok Chun Ng, Wei Wang, Ruichuan Chen, Bo Li
6. Feature Reconstruction Attacks and Countermeasures of DNN Training in Vertical Federated Learning. ***[IEEE TDSC 2024, Pending Major Revision]***
   - Peng Ye, Zhifeng Jiang, Wei Wang, Bo Li, Baochun Li
7. FLASHE: Additively Symmetric Homomorphic Encryption for Cross-Silo Federated Learning. ***[arXiv 2021]***
   - Zhifeng Jiang, Wei Wang, Yang Liu

The publications covered by this thesis is marked with ☆

Thank You!