Flamingo: Multi-Round Single-Server Secure Aggregation with Applications to Private Federated Learning

Yiping Ma[1] Jess Woods[1] Sebastian Angel[1,2] Antigoni Polychroniadou[3] Tal Rabin[1]

[1]University of Pennsylvania

[2]Microsoft Research

[3]J.P. Morgan AI Research & AlgoCRYPT CoE

Data-driven applications nowadays

Service providers collect and analyze user data
in order to provide customized functionalities.

Data-driven applications nowadays

Simply put, to protect users at scale, we rely on machine learning powered by user feedback to catch spam and help us identify patterns in large data sets—making it easier to adapt quickly to ever-changing spam tactics. Gmail employs a number of AI-driven filters that determine what gets marked as spam. These filters look at a variety of signals, including characteristics of the IP address, domains/subdomains, whether bulk senders are authenticated, and user input. User feedback, such as when a user marks a certain email as spam or signals they want a sender's emails in their inbox, is key to this filtering process, and our filters learn from user actions.

Art

Creating a text classifier model

Train a machine learning model to classify natural language text.

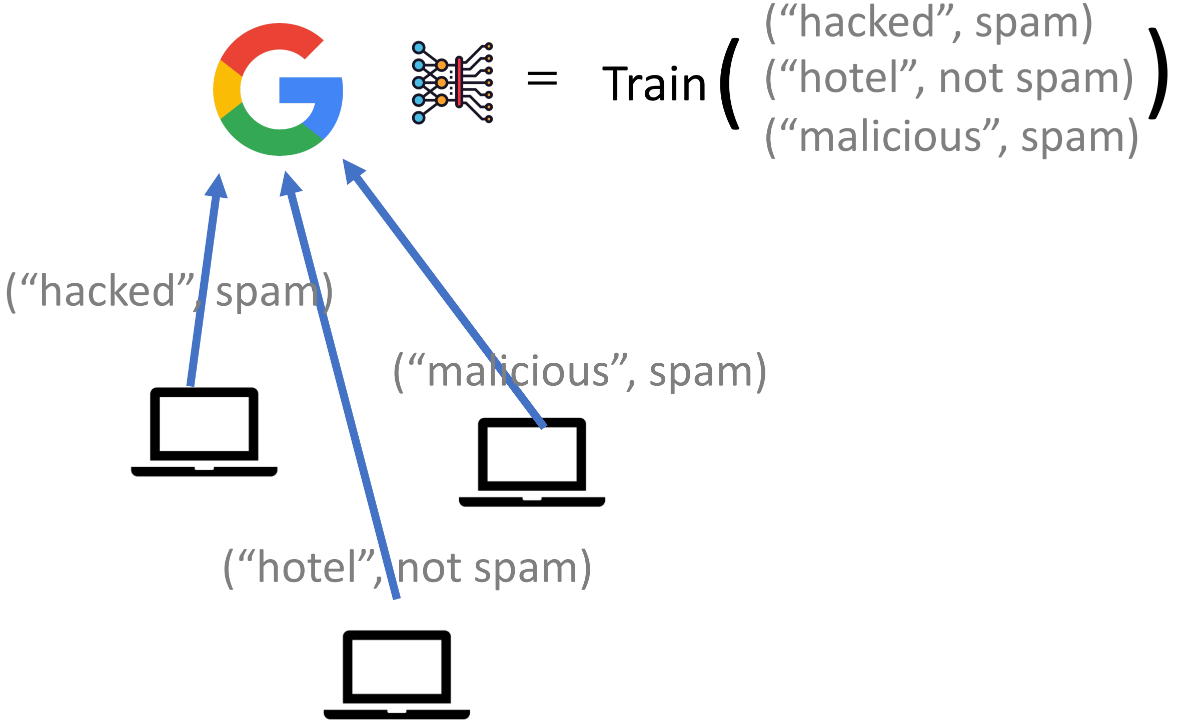
Ama

Elevate the customer experience with ML-powered personalization

Get started with Amazon Personalize

Centralized vs. decentralized training

Centralized



Decentralized

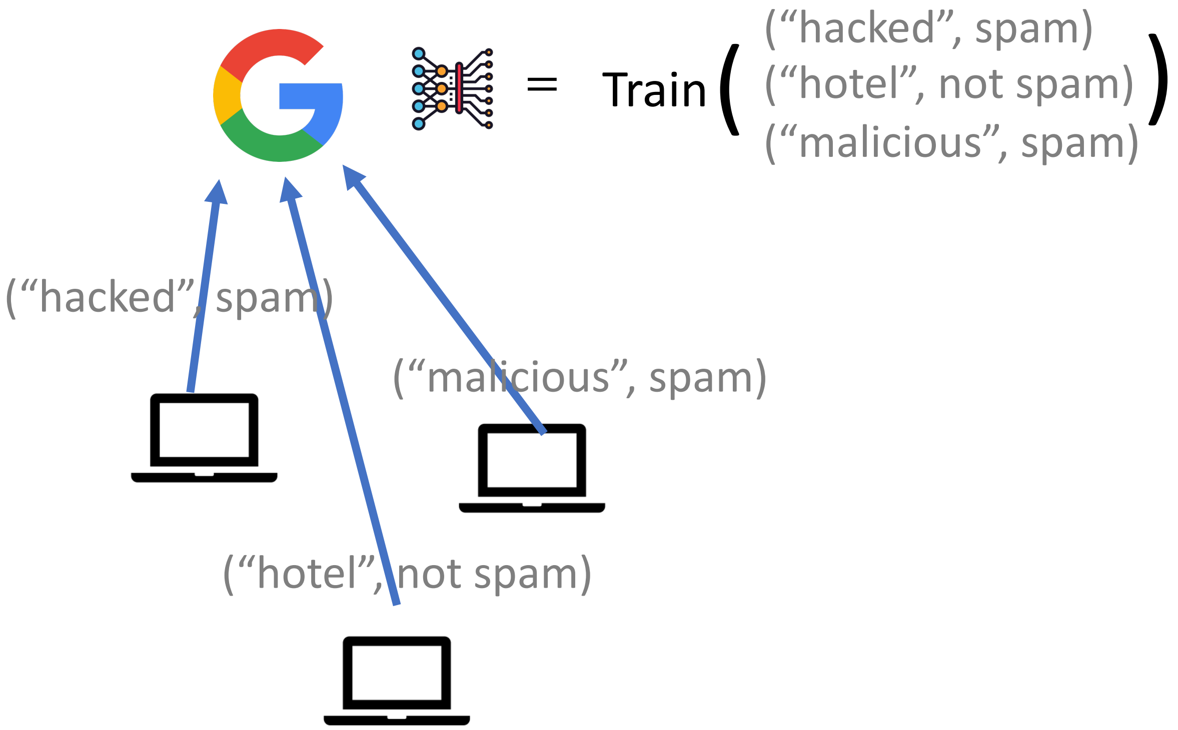
"Federated learning" [McMahan et al. in 2016]

Many clients (users) collaboratively train a model under the orchestration of a central server (service provider).

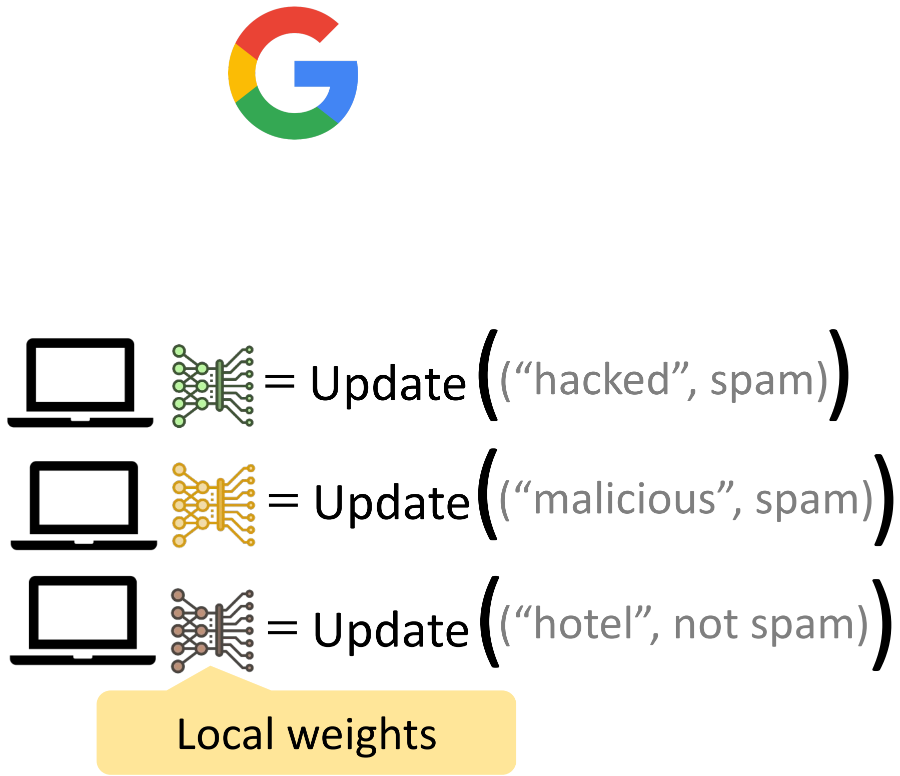
Data never leaves user devices!

Centralized vs. decentralized training

Centralized

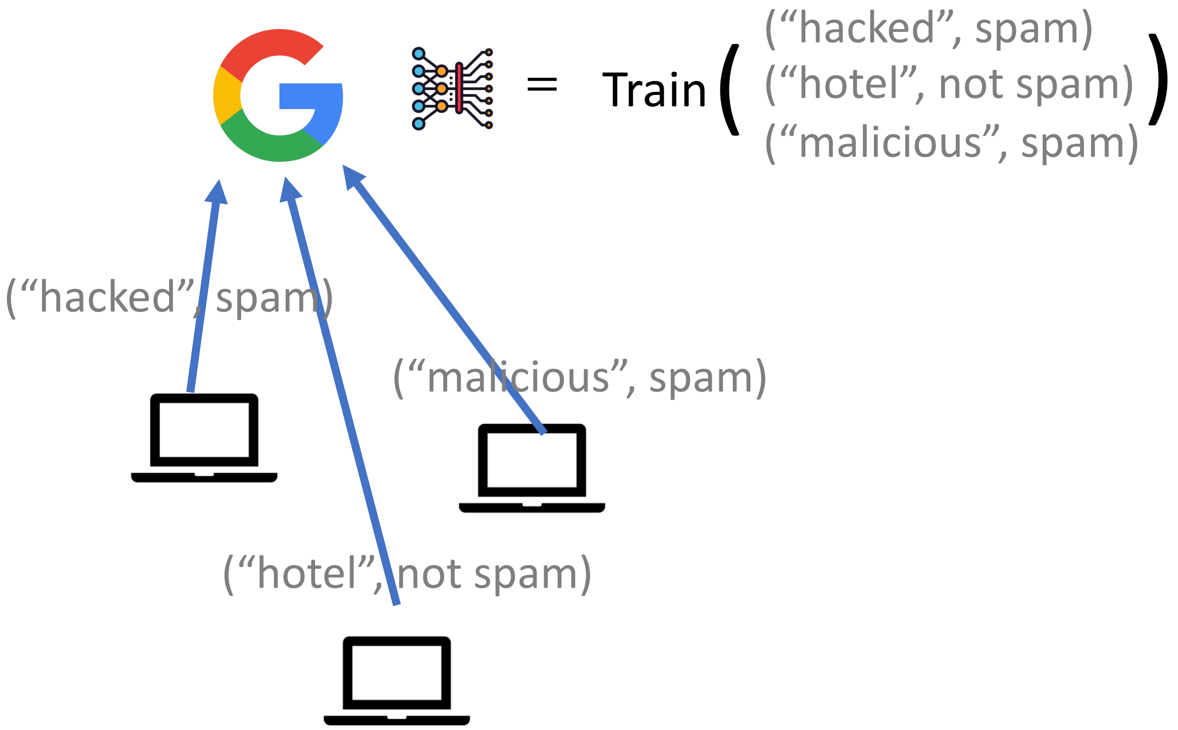


Decentralized

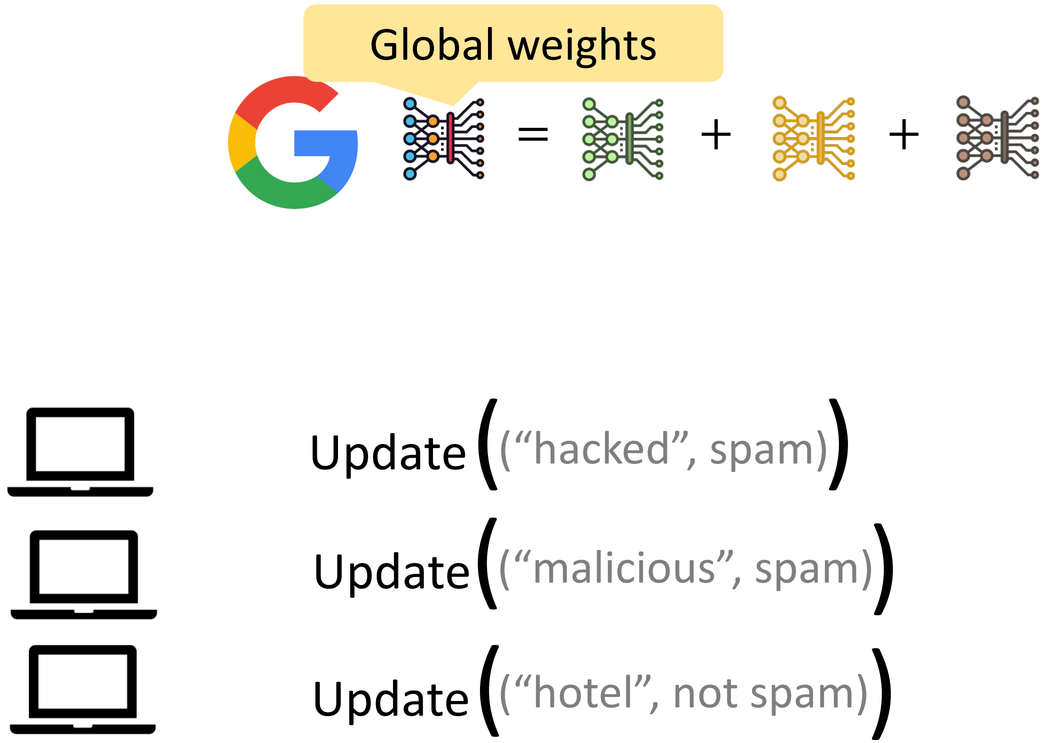


Centralized vs. decentralized training

Centralized



Decentralized



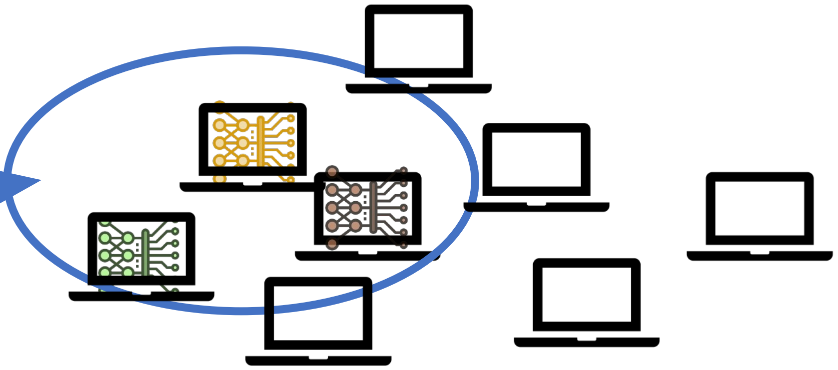
Centralized vs. decentralized training

Centralized	Decentralized

 = Train (("hacked", spam)
("hotel", not spam)
("malicious", spam)
("apple", not spam)
("dog", not spam)
("lottery", spam)
("random", spam))

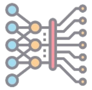
 = + +

A few hundreds to a few thousands of clients

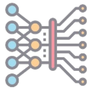


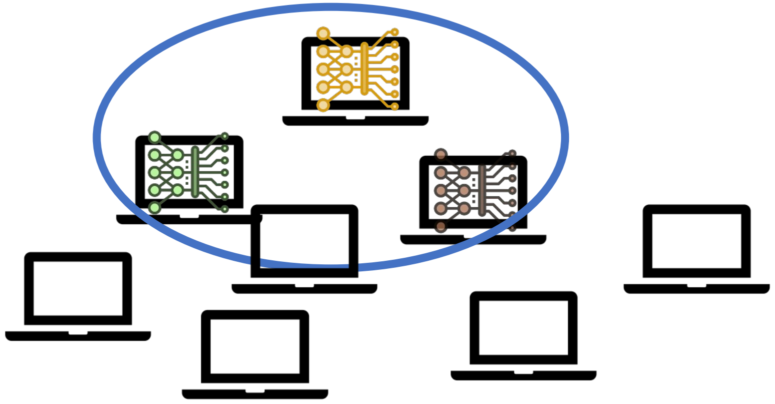
Centralized vs. decentralized training

Centralized

 = Train (
("hacked", spam)
("hotel", not spam)
("malicious", spam)
("apple", not spam)
("dog", not spam)
("lottery", spam)
("random", spam)
)

Decentralized

 = + +



Centralized vs. decentralized training

Centralized

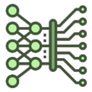
= Train (
("hacked", spam)
("hotel", not spam)
("malicious", spam)
("apple", not spam)
("dog", not spam)
("lottery", spam)
("random", spam)
)

Decentralized

= + +

Federated learning: steps forward

- Weights do not necessarily hide data: inference attack
  [Zhu et al. 2019]

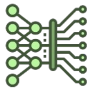


("hacked", spam)

- Training does not need individual weights; only the sum is needed

Federated learning: steps forward

- Weights do not necessarily hide data: inference attack
  [Zhu et al. 2019]

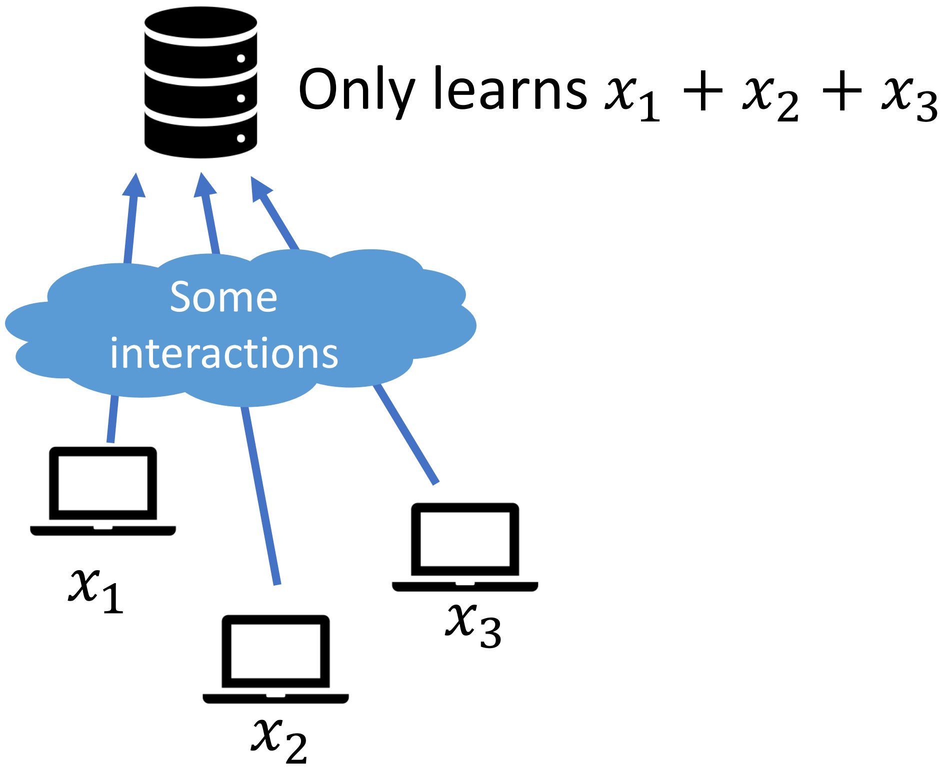


("hacked", spam)

- Training does not need individual weights; only the sum is needed

Secure aggregation for federated learning

- Secure aggregation (A special case of MPC [Yao 1986])

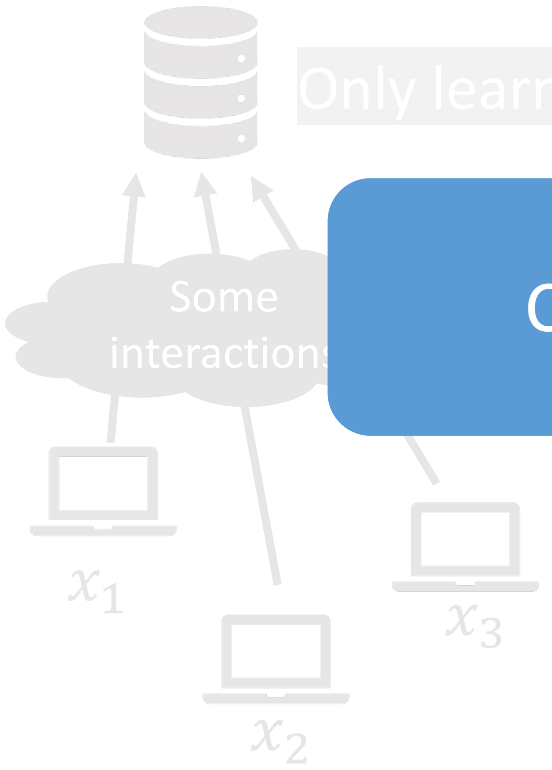


Many works under different communication models, cryptographic assumptions, etc.

- Secret sharing
  [KRKR 2020], [DSQG+ 2022], …
- Threshold homomorphic encryption
  [SGA 2021], [SHYL+ 2022], …
- Pairwise masking
  [BIKM+ 2017], [BBGLR 2020], …
- Other styles
  [SSVR+ 2022], [GPSBB 2022], …

Secure aggregation for federated learning

- Secure aggregation (A special case of MPC [Yao 1986])



Only learns $x_1 + x_2 + x_3$

Many works under different communication models, cryptographic assumptions, etc.

Concrete efficiency but not asymptotics

[SGA 2021], [SHYL+ 2022], …

- Pairwise masking
  [BIKM+ 2017], [BBGLR 2020], …
- Other styles
  [SSVR+ 2022], [GPSBB 2022], …

Federated learning has complex setting

- From the federation side—restricted clients (mobile devices)
  - Limited computation power
  - Unstable network connection

- From the machine learning side—large parameters
  - Inputs: model weights, e.g., ~500K in popular models for CIFAR100
  - Participants: 100-5000 per iteration
  - Training: many iterations, e.g., ~300 for CIFAR100

Federated learning has complex setting

- From the federation side—restricted clients (mobile devices)
  - Limited computation power
  - Unstable network connection

Lightweight client computation

Tolerate dropouts at any point

- From the machine learning side—large parameters
  - Inputs: model weights, e.g., ~500K in popular models for CIFAR100
  - Participants: 100-5000 per iteration
  - Training: many iterations, e.g., ~300 for CIFAR100

Federated learning has complex setting

- From the federation side—restricted clients (mobile devices)
  - Limited computation power
  - Unstable network connection

- From the machine learning side—large parameters
  - Inputs: model weights, e.g., ~500K in popular models for CIFAR100
  - Participants: 100-5000 per iteration
  - Training: many iterations, e.g., ~300 for CIFAR100

Reasonable efficiency

Prior designs are not the best fit for a full training

- From the federation side—restricted clients (mobile devices)
  - Limited computation power
  - Unstable network connection

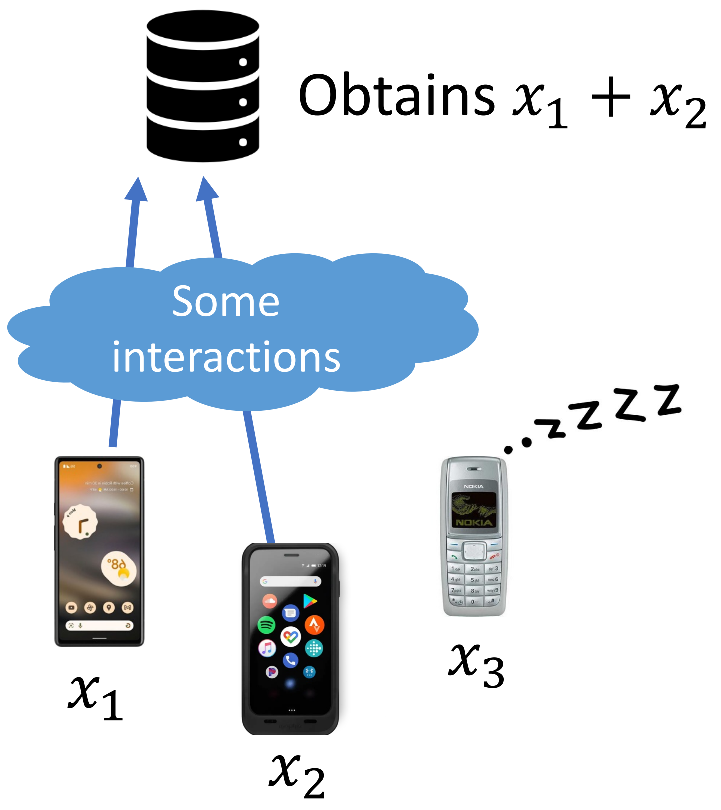
- From the machine learning side—large parameters
  - Inputs: model weights, e.g., ~500K in popular models for CIFAR100
  - Participants: 100-5000 per iteration
  - Training: many iterations, e.g., ~300 for CIFAR100

One summation: multiple round trips, some of which are expensive

Having fewer round trips is important

- Reduce bias and improve quality

- Reduce run time
  Will discuss in evaluation section
  why round trips matter a lot



We propose Flamingo

- From the federation side—restricted clients (mobile devices)
  - Limited computation power
  - Unstable network connection

Lightweight client computation

Tolerate dropouts at any point

- From the machine learning side—large parameters
  - Inputs: model weights, e.g., ~500K in popular models for
  - Participants: 100-5000 per iteration
  - Training: many iterations, e.g., ~300 for CIFAR100

Can practically run for a full training session

Same threat model as in prior work: a malicious adversary controlling the server and a subset of the clients

Flamingo has two key ideas

- A fault-tolerant private sum protocol
  based on pairwise secrets and threshold decryption

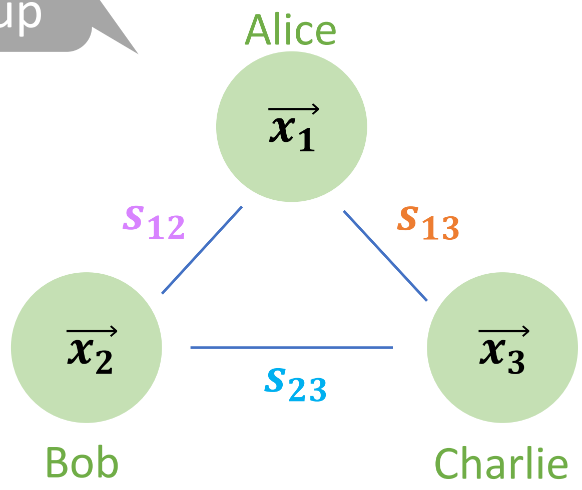
- A way to reuse pairwise secrets over many iterations

A fault-tolerant private sum protocol

Pairwise secrets

BIKM+ 2017,
BBGLR 2020

Take some cost
to set them up



$$\vec{v_1} = \vec{x_1} + \mathbf{PRG}\,(s_{12}) + \mathbf{PRG}(s_{13})$$

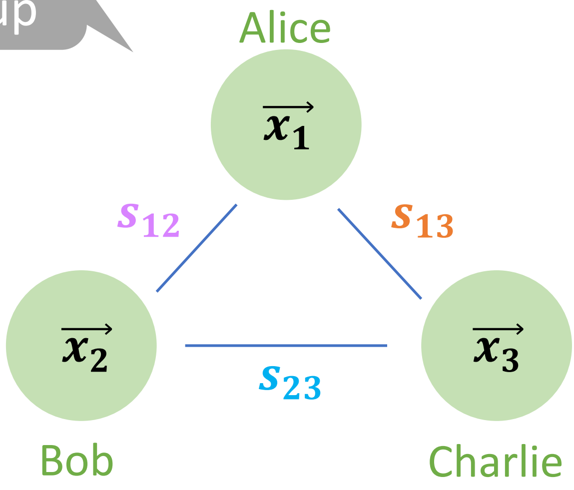
Semi-honest

A fault-tolerant private sum protocol

Pairwise secrets

BIKM+ 2017, BBGLR 2020

Take some cost to set them up



$$\vec{v_1} = \vec{x_1} + \cancel{\mathrm{PRG}(s_{12})} + \cancel{\mathrm{PRG}(s_{13})}$$

$$\vec{v_2} = \vec{x_2} - \cancel{\mathrm{PRG}(s_{12})} + \cancel{\mathrm{PRG}(s_{23})}$$

$$\vec{v_3} = \vec{x_3} - \cancel{\mathrm{PRG}(s_{13})} - \cancel{\mathrm{PRG}(s_{23})}$$

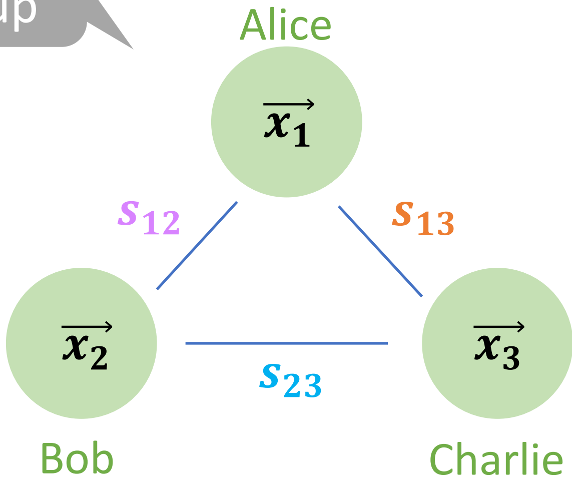
A fault-tolerant private sum protocol

Pairwise secrets

BIKM+ 2017, BBGLR 2020

$$\sum_{i=1}^{3} \vec{v_i} = \sum_{i=1}^{3} \vec{x_i}$$

Take some cost to set them up

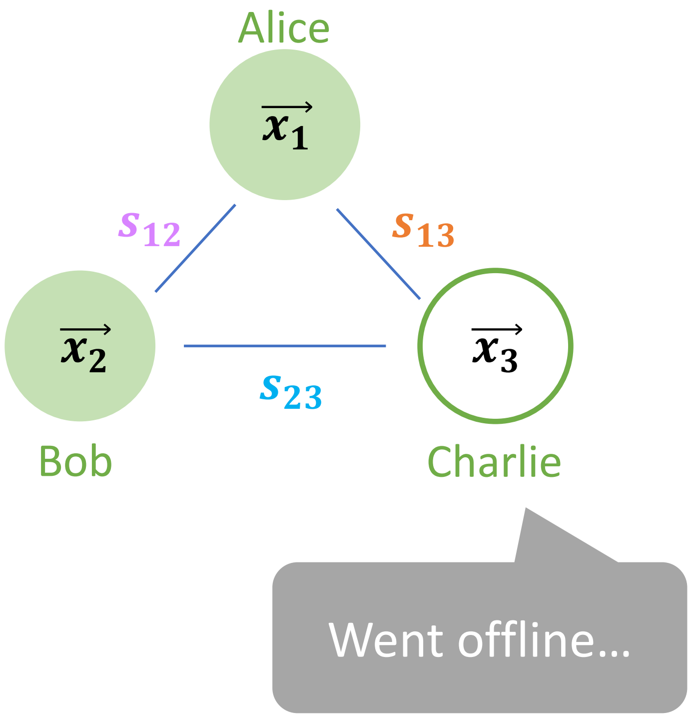


$$\begin{aligned}
\vec{v_1} &= \vec{x_1} + \cancel{\mathrm{PRG}(s_{12})} + \cancel{\mathrm{PRG}(s_{13})} \\
\vec{v_2} &= \vec{x_2} - \cancel{\mathrm{PRG}(s_{12})} + \cancel{\mathrm{PRG}(s_{23})} \\
\vec{v_3} &= \vec{x_3} - \cancel{\mathrm{PRG}(s_{13})} - \cancel{\mathrm{PRG}(s_{23})}
\end{aligned}$$

Efficient despite large inputs

A fault-tolerant private sum protocol

Pairwise secrets



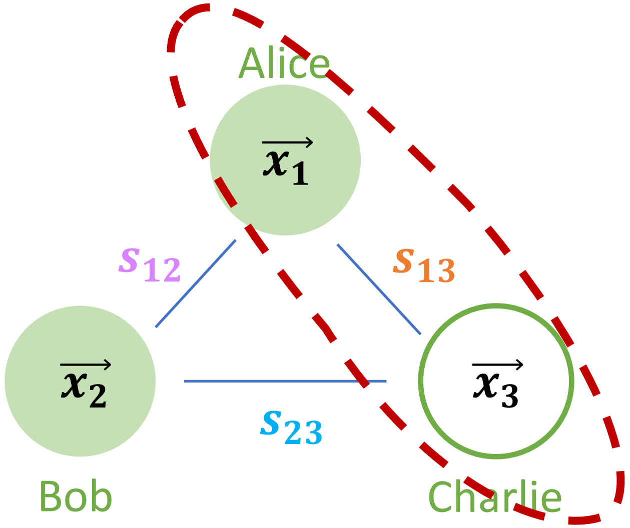
$$\vec{v_1} = \vec{x_1} + \cancel{\mathrm{PRG}(s_{12})} + \mathrm{PRG}(s_{13})$$

$$\vec{v_2} = \vec{x_2} - \cancel{\mathrm{PRG}(s_{12})} + \mathrm{PRG}(s_{23})$$

Reveal the secrets to the server!

A fault-tolerant private sum protocol

Threshold decryption



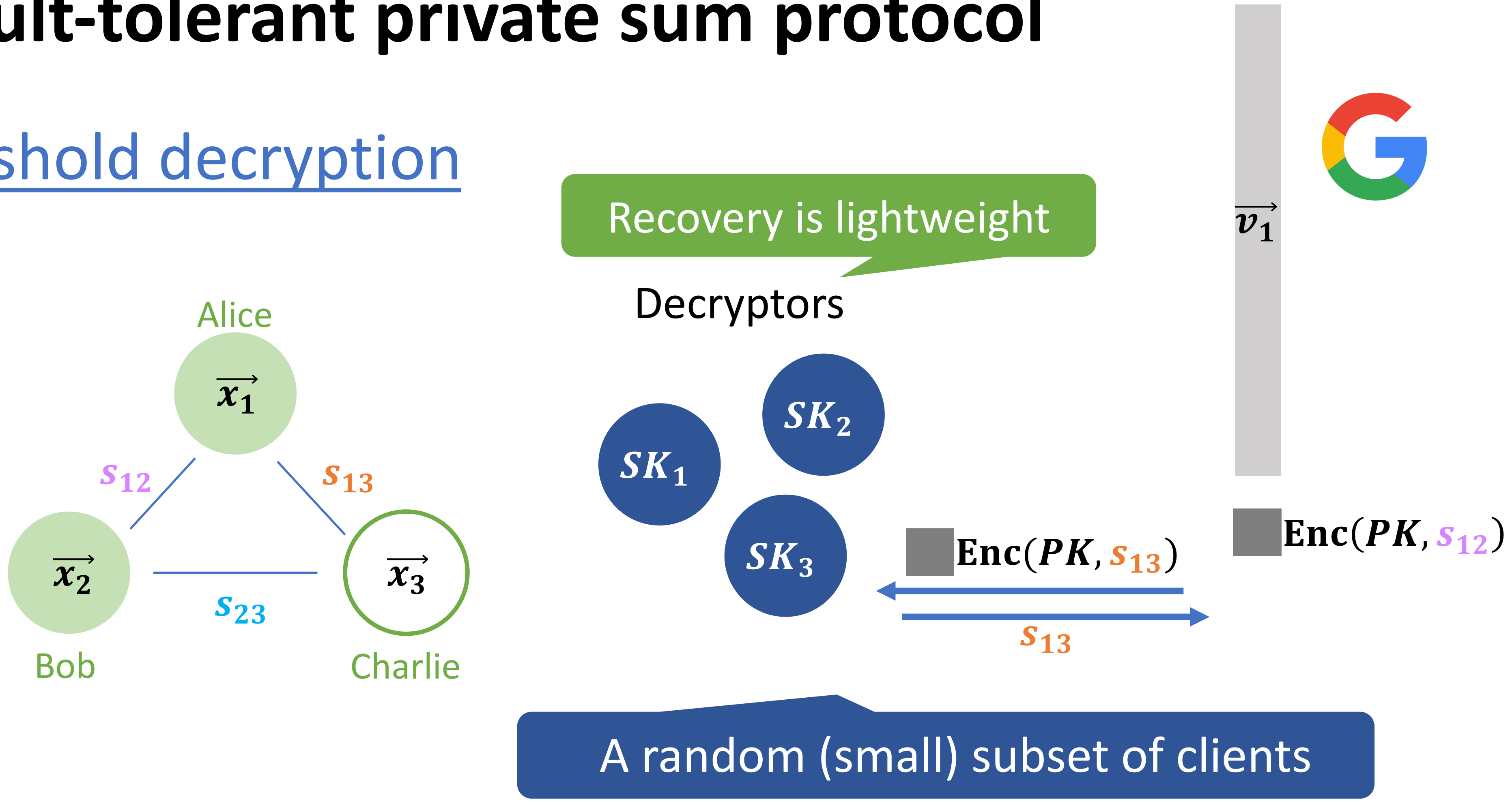
$$\vec{v_1} = \vec{x_1} + \mathrm{PRG}\,(s_{12}) + \mathrm{PRG}(s_{13})$$

$\mathrm{Enc}(PK, s_{12})$

$\mathrm{Enc}(PK, s_{13})$

A fault-tolerant private sum protocol

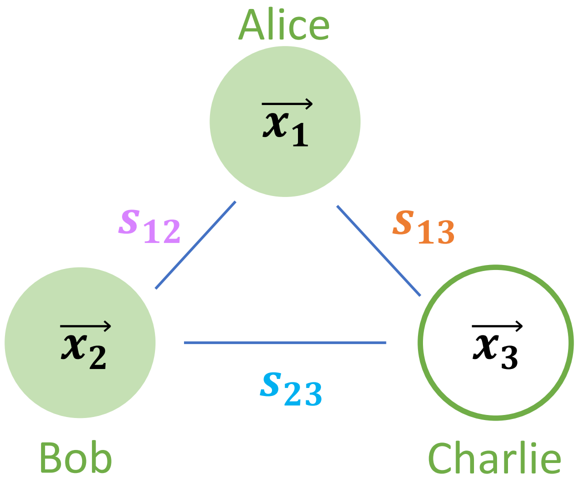
Threshold decryption



Semi-honest

A fault-tolerant private sum protocol

Threshold decryption



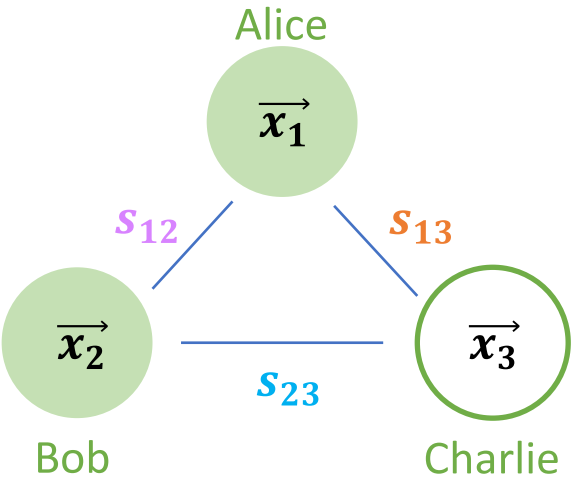
$\vec{v_1}$ $\vec{v_2}$ s_{13} s_{23}

$$\vec{v_1} = \vec{x_1} + \mathbf{PRG}(s_{12}) + \mathbf{PRG}(s_{13})$$

$$\vec{v_2} = \vec{x_2} - \mathbf{PRG}(s_{12}) + \mathbf{PRG}(s_{23})$$

A fault-tolerant private sum protocol

Threshold decryption



$$\vec{v_1} = \vec{x_1} + \mathrm{PRG}\,(s_{12}) + \mathrm{PRG}(s_{13})$$

$$\vec{v_2} = \vec{x_2} - \mathrm{PRG}\,(s_{12}) + \mathrm{PRG}(s_{23})$$

$$- \mathrm{PRG}\,(s_{13}) - \mathrm{PRG}(s_{23})$$

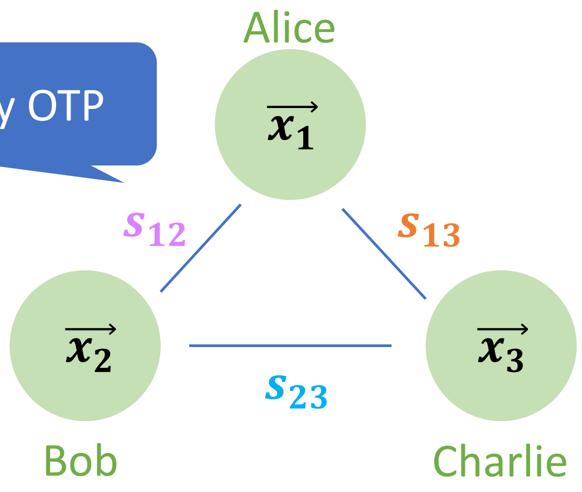
$$\vec{v_1} + \vec{v_2} - \mathrm{PRG}\,(s_{13}) - \mathrm{PRG}(s_{23}) = \vec{x_1} + \vec{x_2}$$

Semi-honest

Reusing the secrets

Simple idea, but cannot work for [BBGLR 2020] due to a crucial design difference for fault tolerance

Essentially OTP



$$\vec{v}_1 = \vec{x}_1 + \mathbf{PRG}\,(s_{12}^t) + \mathbf{PRG}(s_{13}^t)$$

$\mathbf{Enc}(PK, s_{12}^t)$

$\mathbf{Enc}(PK, s_{13}^t)$

Iteration t: $s_{12}^t = \mathrm{PRF}(s_{12}, t)$

With the two key ideas →

Do the costly setup once,
and run the lightweight sum many times

More details in the paper

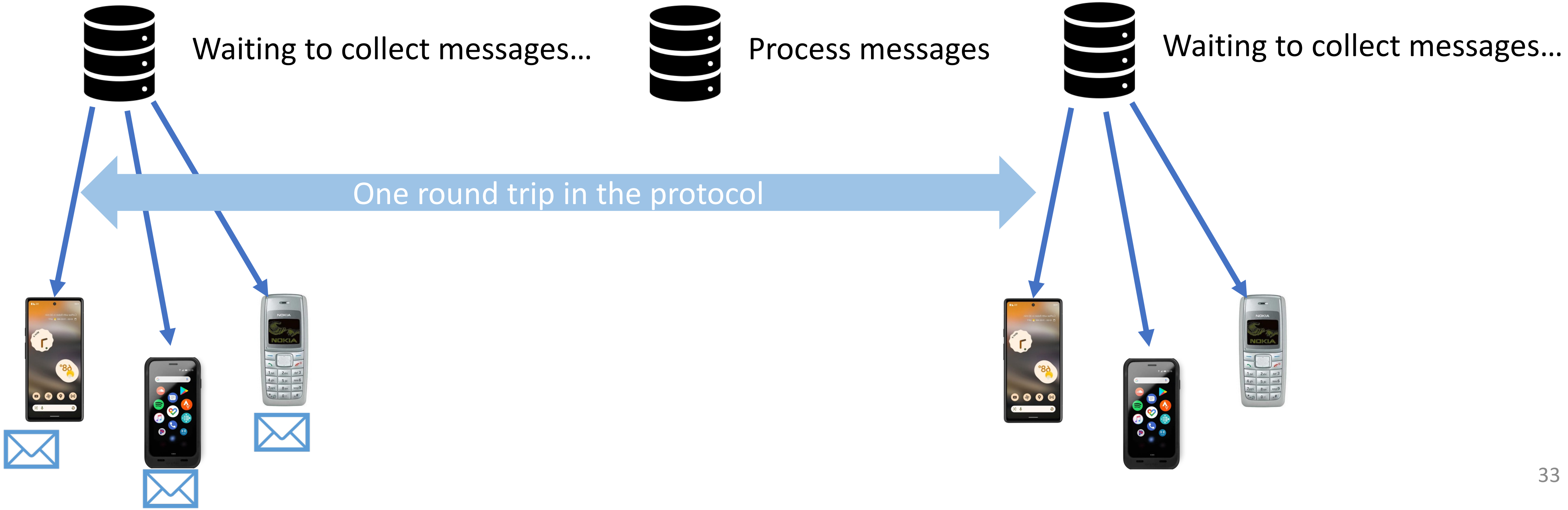
- How decryptors work
  - Selection
  - Sharing of SK
  - Switching decryptors over time
- How setup is done
- How to achieve malicious security
- Efficient instantiation of cryptographic primitives, system-level optimizations

Evaluation results

- What is the right factor to look at?
  - Computation cost was the focus: [BIKM+ 2017] → [BBGLR 2020]
  - When computation is made cheap, what matters is the “waiting time”

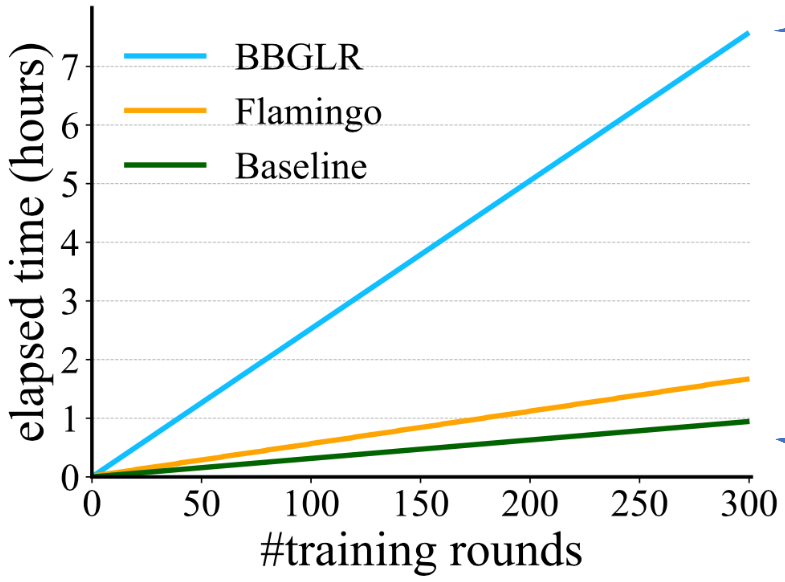
Evaluation results

- What is the right factor to look at?
  - Computation cost was the focus: [BIKM+ 2017] → [BBGLR 2020]
  - When computation is made cheap, what matters is the “waiting time”



Evaluation results

- Feasibility of training a neural network on CIFAR100
- Simulation using a multi-agent messaging system ABIDES



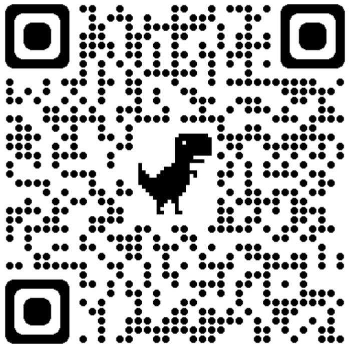
6 RTTs, with all clients

3 RTTs, only 1 RTT with all clients

1 RTT with all clients

(malicious version)

ABIDES



Summary

- This work: A secure aggregation system that handles real-world federated training tasks
- Many interesting future directions
  - Validation of client inputs
  - Stronger security, e.g., adaptive adversary

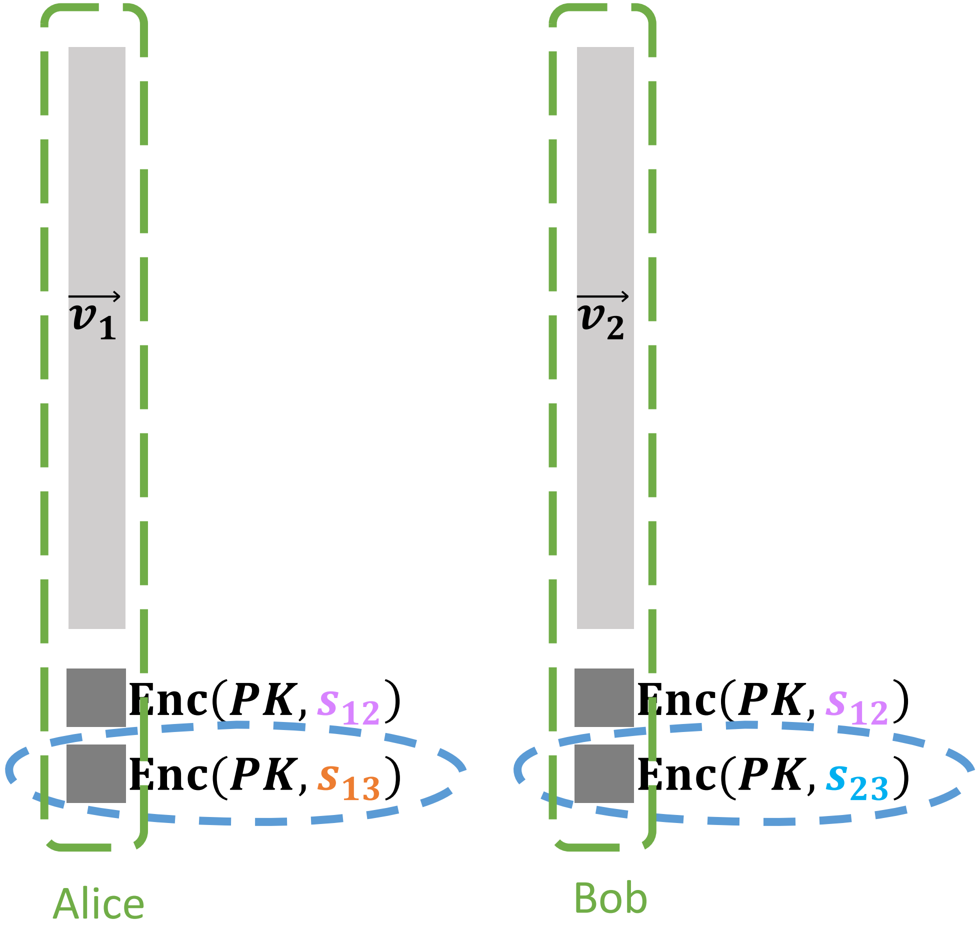
paper

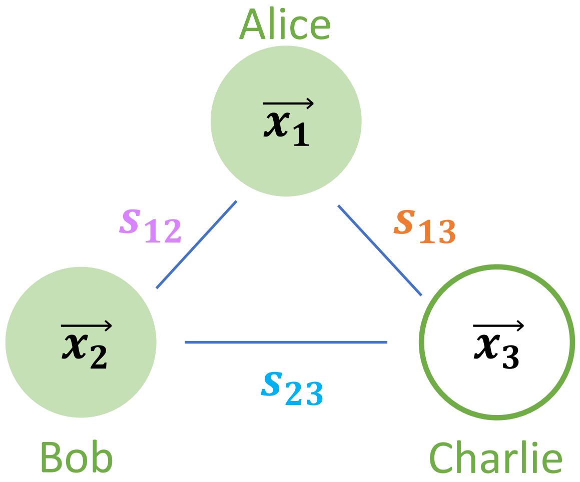
code

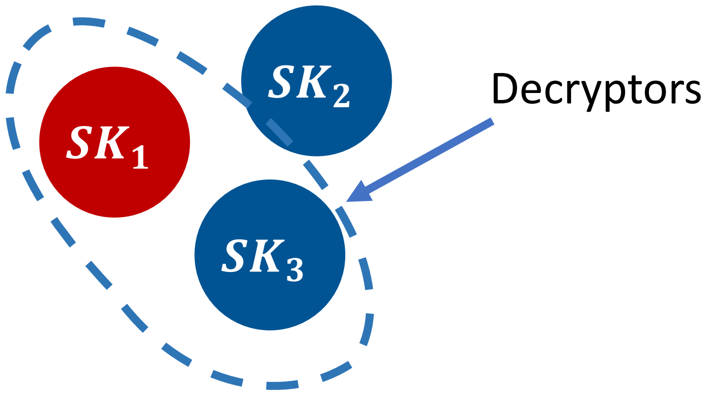
Thanks!

Backup Slides

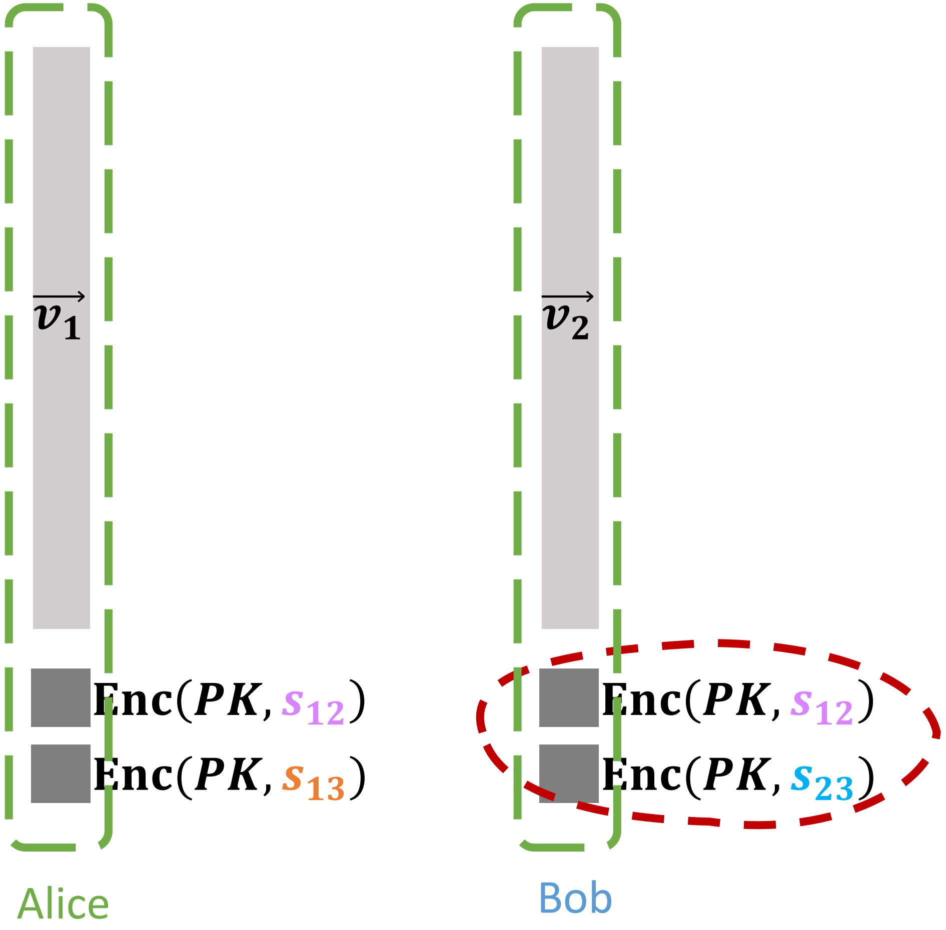
Malicious security

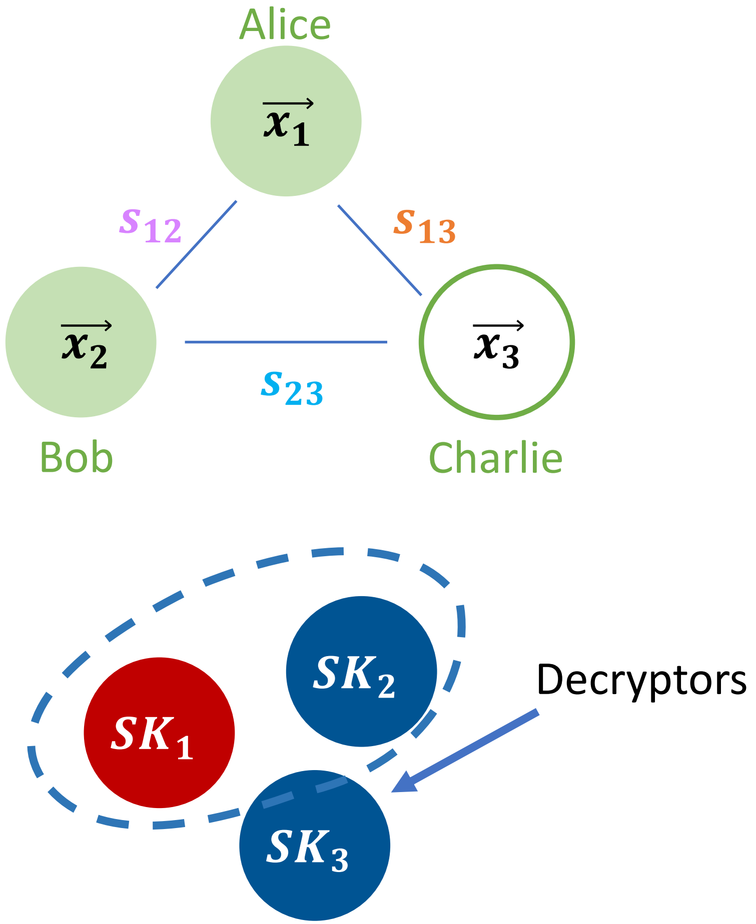




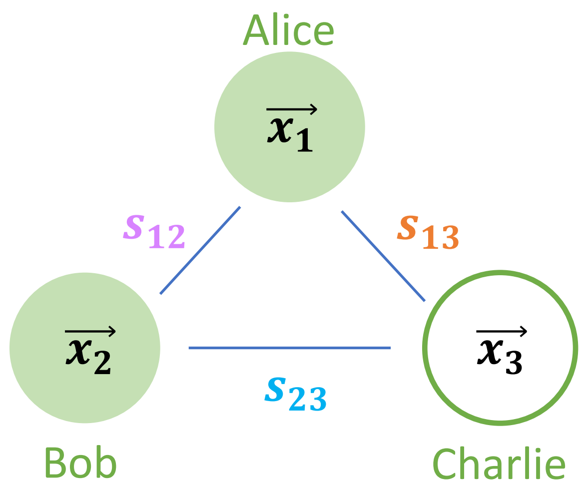


Malicious security



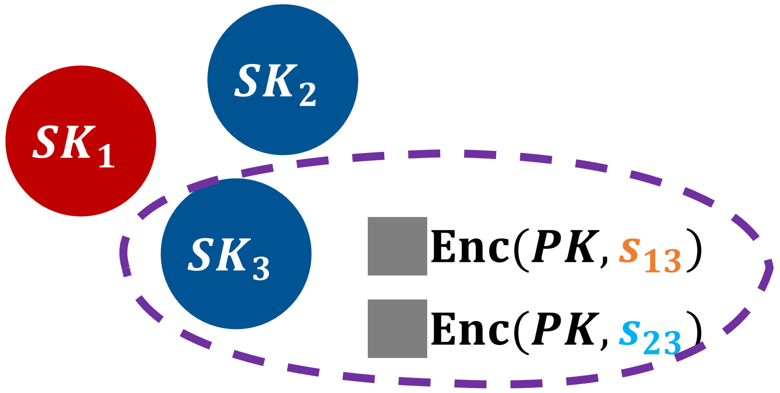


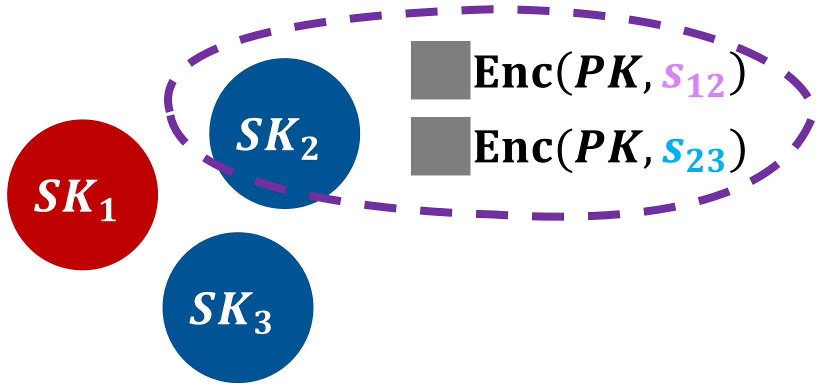
A cross-check round



Key idea:

Honest decryptors agree on what to decrypt





A fault-tolerant sum with malicious security