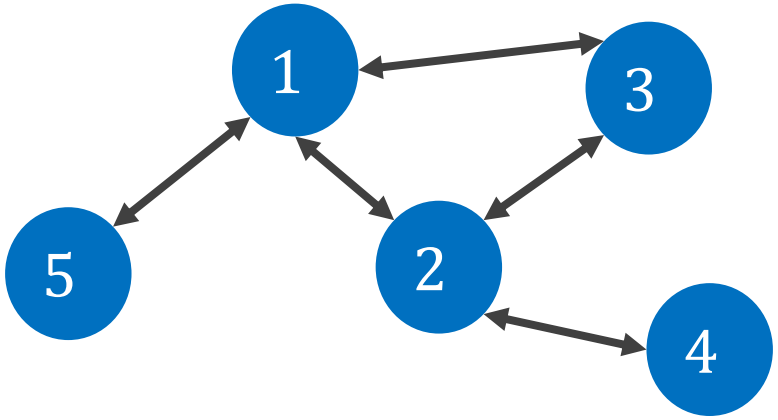
Tutorial: Part V

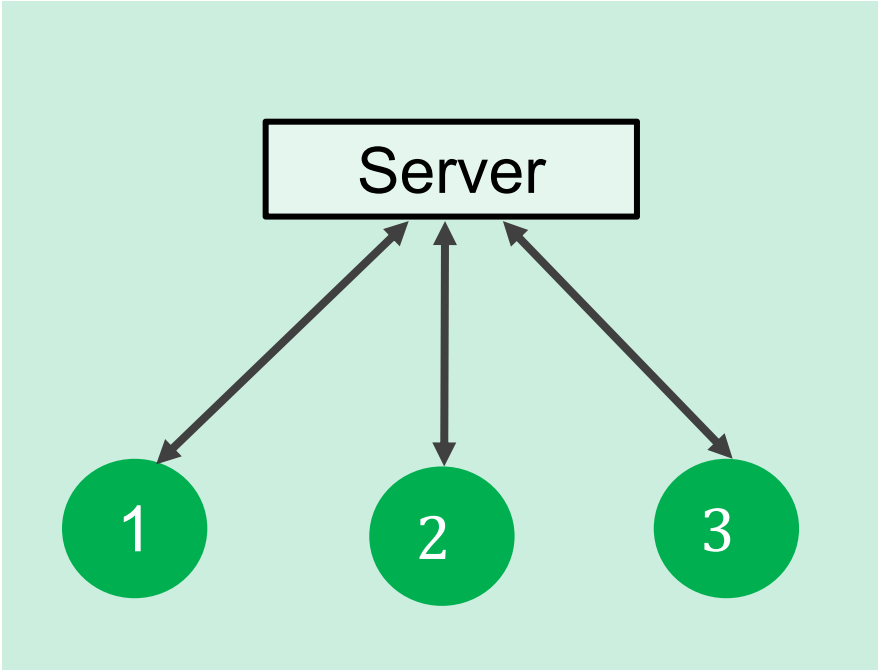
# Security and Privacy in Distributed Optimization and Learning

Nitin Vaidya
Georgetown University

# Privacy

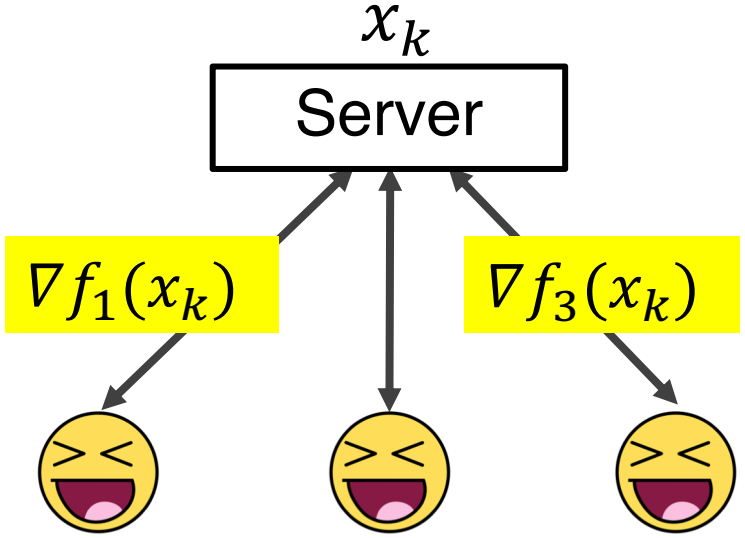
# Architectures



# Distributed Optimization

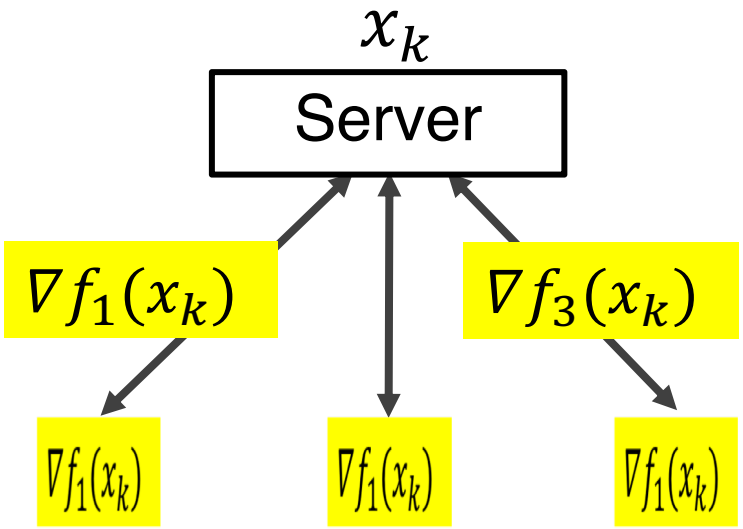
- Server maintains estimate $x_k$



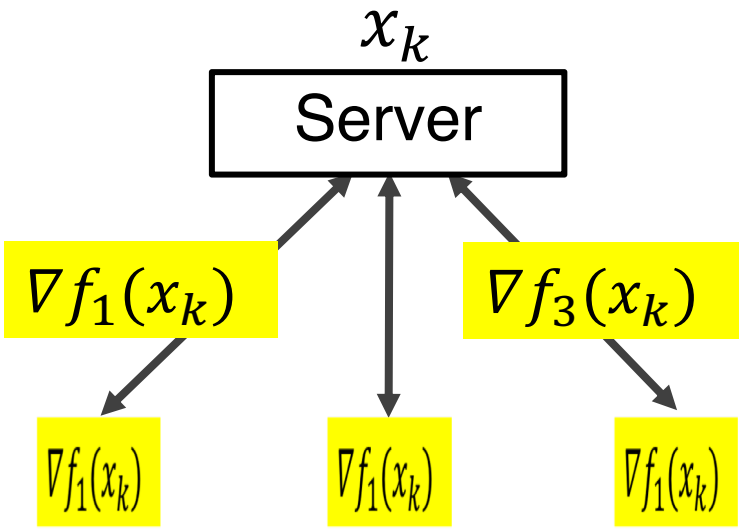

In each iteration

- Each agent $i$
  - Receives $x_k$ from server
  - Uploads gradient $\nabla f_i(x_k)$
- Server updates estimate

$$x_{k+1} \longleftarrow x_k - \lambda_k \sum \nabla f_i(x_k)$$




Server observes gradients ➔ privacy compromised




Server observes gradients ➔ privacy compromised

Achieve privacy and yet collaboratively optimize

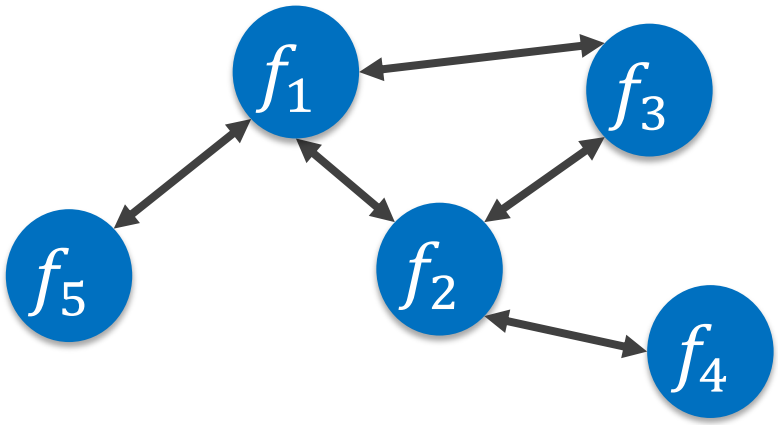
# Decentralized Optimization

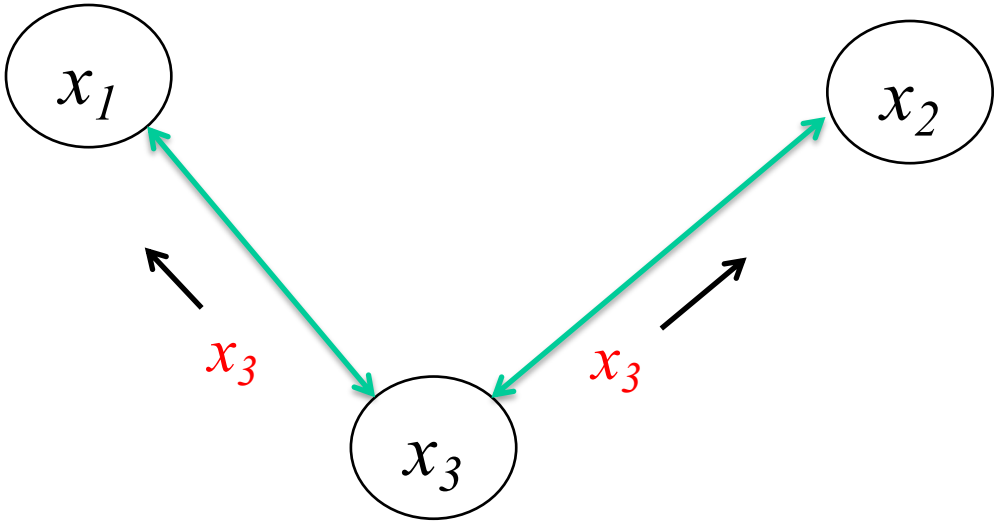
- Each agent maintains local estimate $x$

In each iteration

- Compute weighted average with neighbors' estimates
- Apply own gradient to own estimate

- Local estimates converge to $argmin \sum_i f_i(x)$



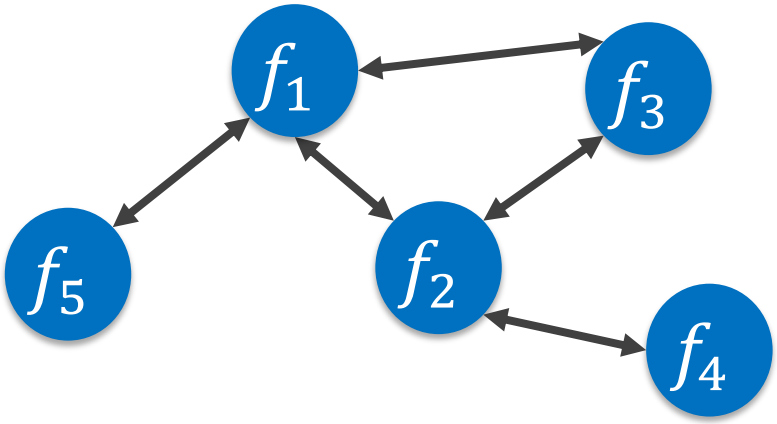



$$x_1[t+1] = \frac{2}{3}x_1[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_1(x_1[t])$$

$$x_3[t+1] = \frac{1}{3}x_1[t] + \frac{1}{3}x_2[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_3(x_3[t])$$

# Peer-to-Peer Architecture

- Neighbors can potentially learn information about an agent's cost function



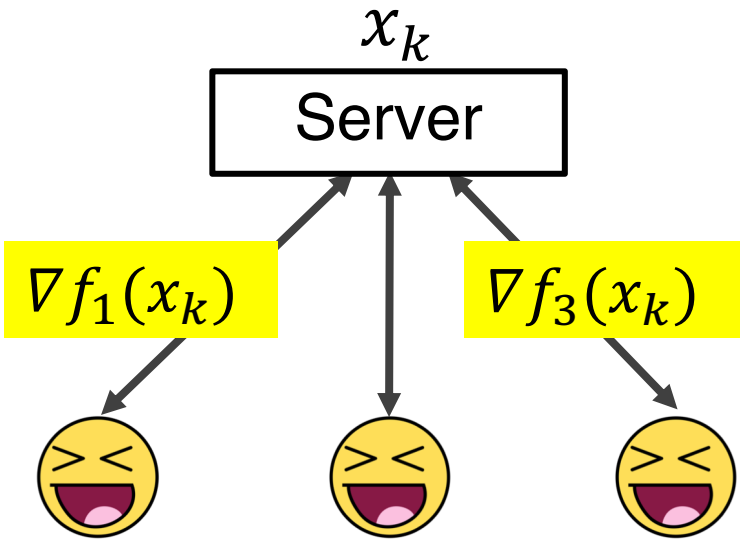
# Approaches for Privacy

# Approaches for Privacy

- Cryptographic methods
- Differential privacy
- Function transformation

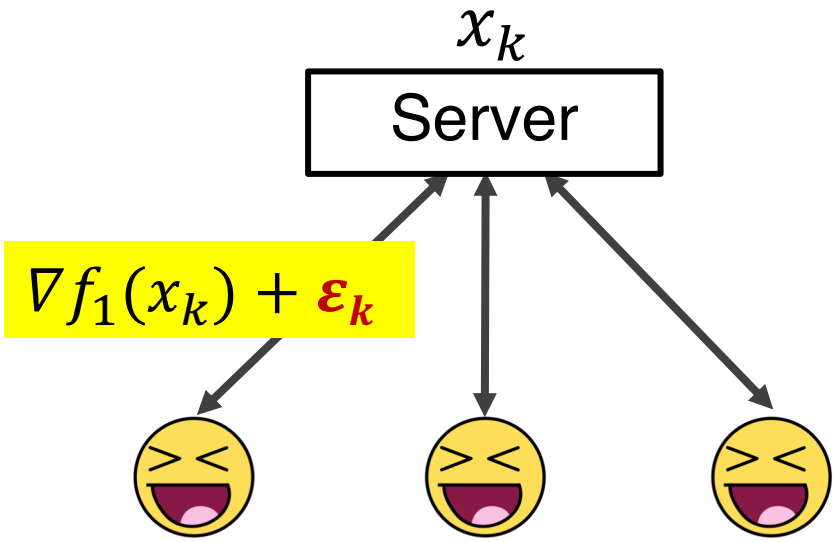
# Encryption

- Basic idea: Agents send encrypted information to the server (or peers)
- Server performs computation on encrypted data
- Need practical/efficient encryption mechanisms



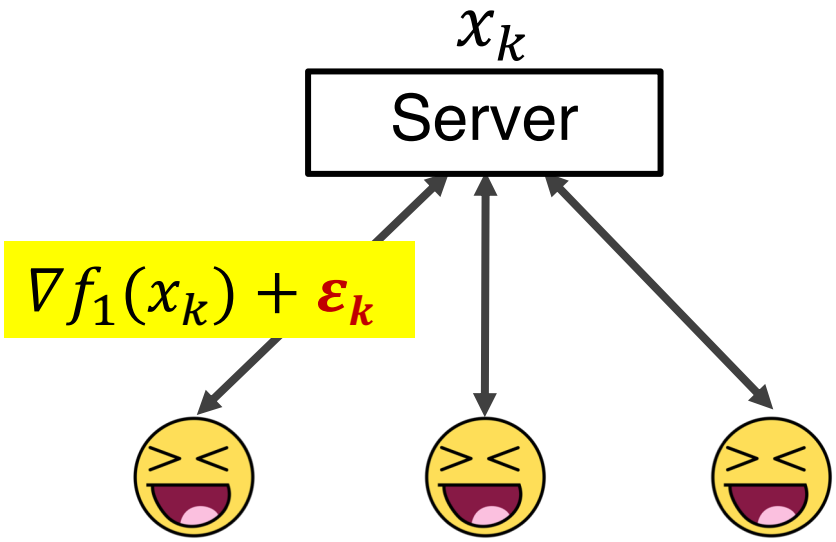
# Differential Privacy
[Shokri, Shmatikov 2015]

- Agents add Laplace distributed noise to the gradients



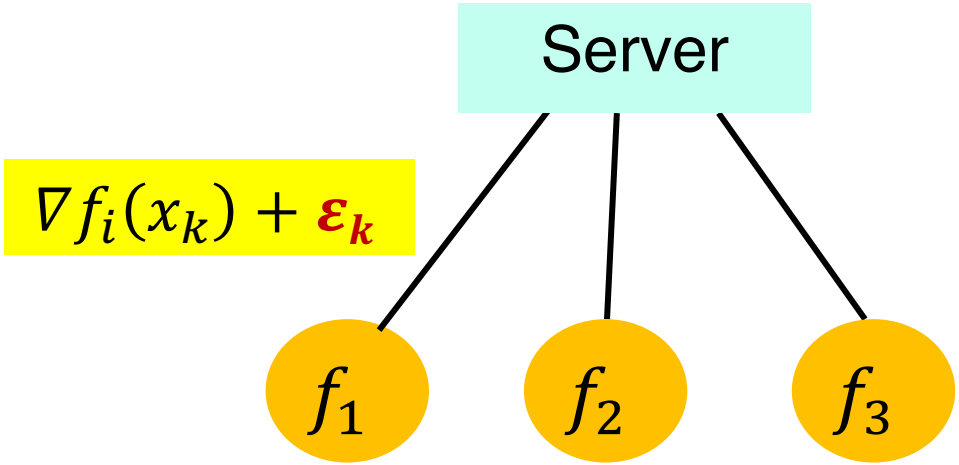
# Differential Privacy for Machine Learning
[Shokri, Shmatikov 2015]

- Agents add Laplace distributed noise to the gradients




Goal: Make it difficult to determine if a particular data item was used during training

# Differential Privacy




Trade-off privacy with accuracy

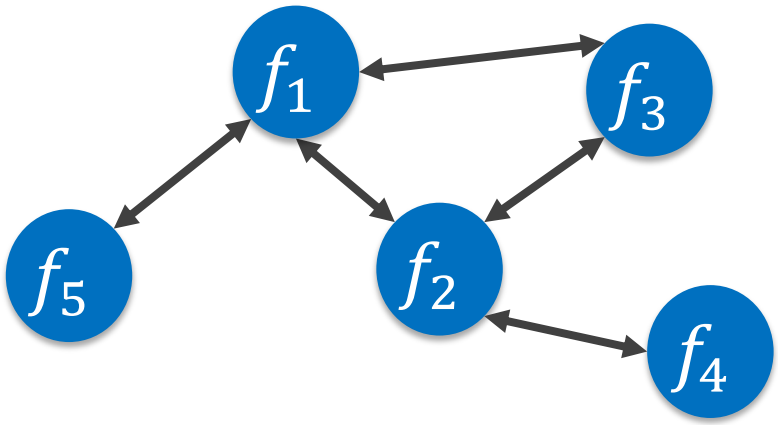
# Decentralized Optimization

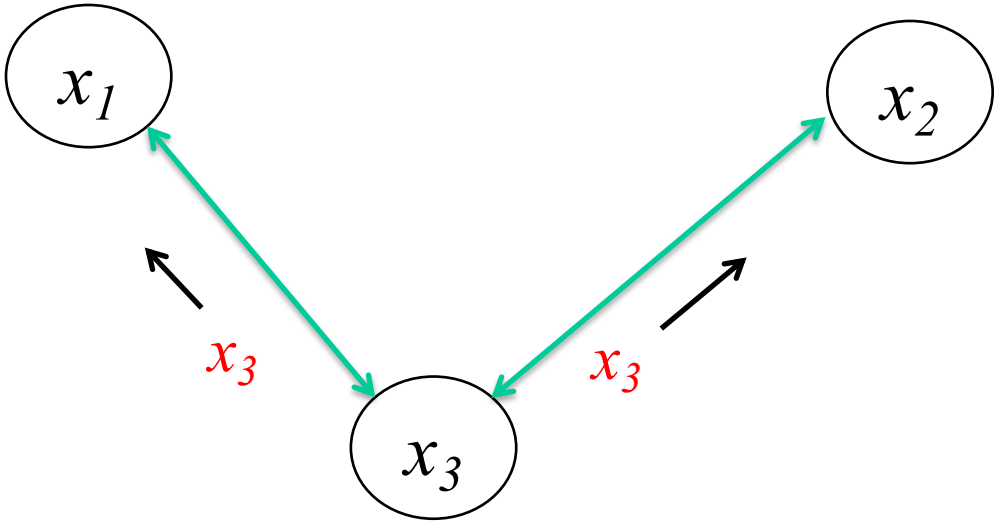
- Each agent maintains local estimate $x$

In each iteration

- Compute weighted average with neighbors' estimates
- Apply own gradient to own estimate

- Local estimates converge to $argmin \sum_i f_i(x)$






$$x_1[t+1] = \frac{2}{3}x_1[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_1(x_1[t])$$

$$x_3[t+1] = \frac{1}{3}x_1[t] + \frac{1}{3}x_2[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_3(x_3[t])$$

# Differential Privacy for Peer-to-Peer Architecture [Huang, Mitra, Vaidya 2015 , 2014]

- Agents add noise to their estimate before sharing with the neighbors

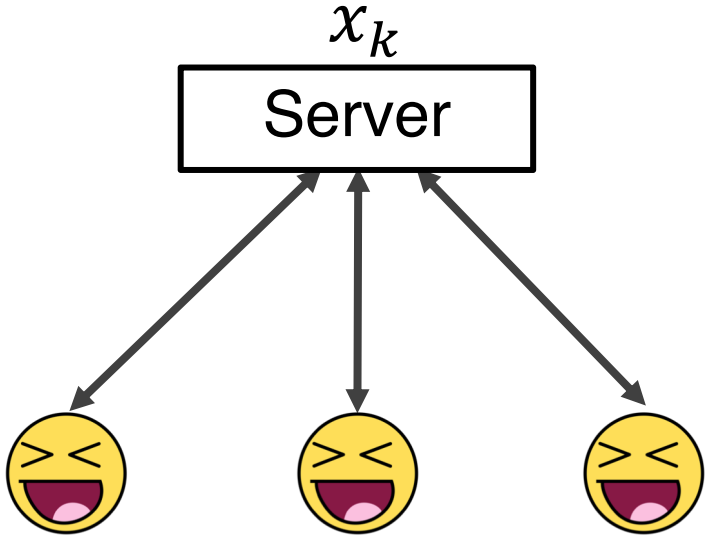
# Other Work

- Several other papers on differential privacy for optimization (such as Dobbe et al. 2020, Showkatbakhsh et al. 2015)

# Federated Architecture

[Kairouz et al 2018]

- Server maintains estimate $x_k$



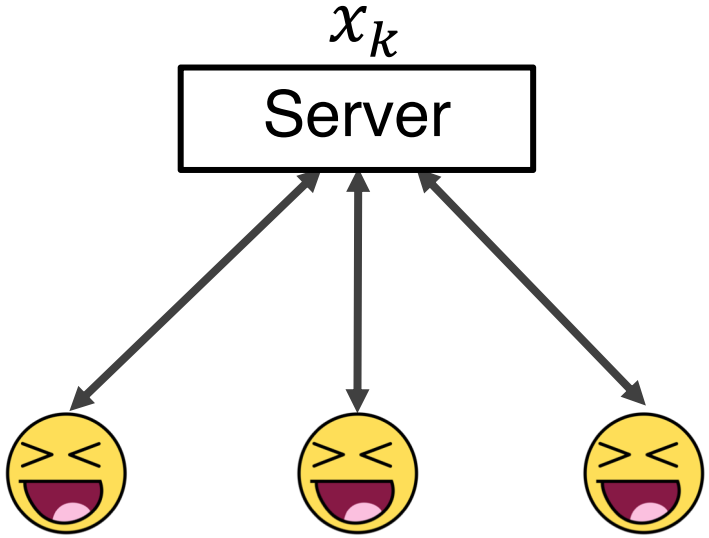
# Federated Architecture

[Kairouz et al 2018]

- Server maintains estimate $x_k$

In each iteration

- Each agent $i$
  - Receive $x_k$ from server

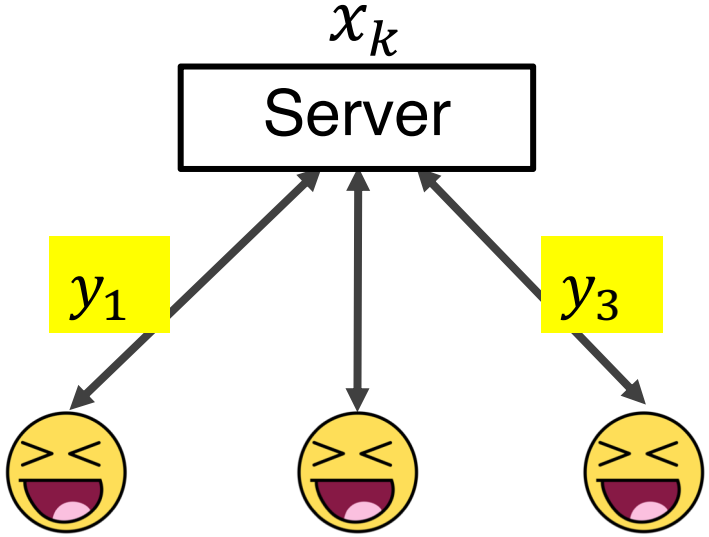


# Federated Architecture

- Server maintains estimate $x_k$

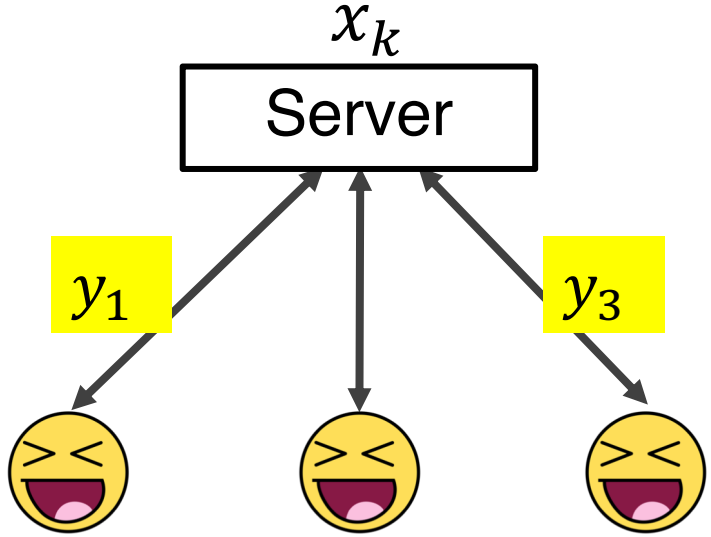
<u>In each iteration</u>

- Each agent $i$
  - Receive $x_k$ from server
  - Compute $y_k = x_k - \lambda_k \nabla f_i(x_k)$
  - Send $y_k$ to server



# Federated Architecture

- Server maintains estimate $x_k$




In each iteration

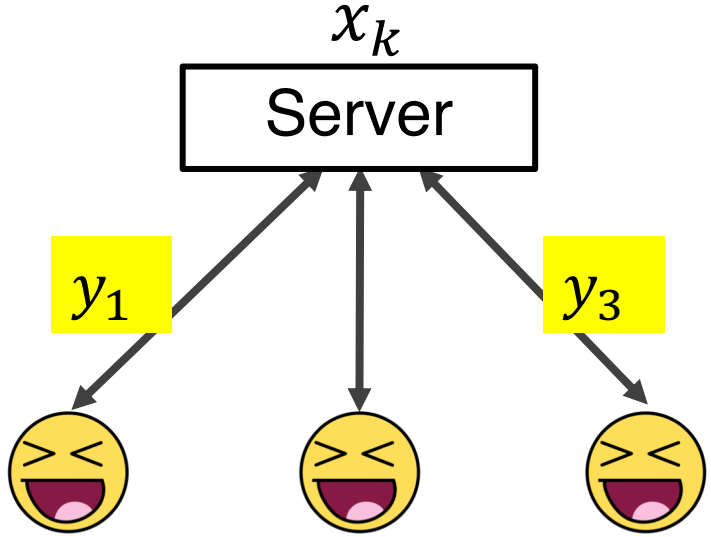
- Each agent $i$
  - Receive $x_k$ from server
  - Compute $y_k = x_k - \lambda_k \nabla f_i(x_k)$
  - Send $y_k$ to server

- Server updates estimate

$$x_{k+1} \leftarrow \frac{1}{n} \sum y_i$$

# Federated Architecture

- Server maintains estimate $x_k$




In each iteration

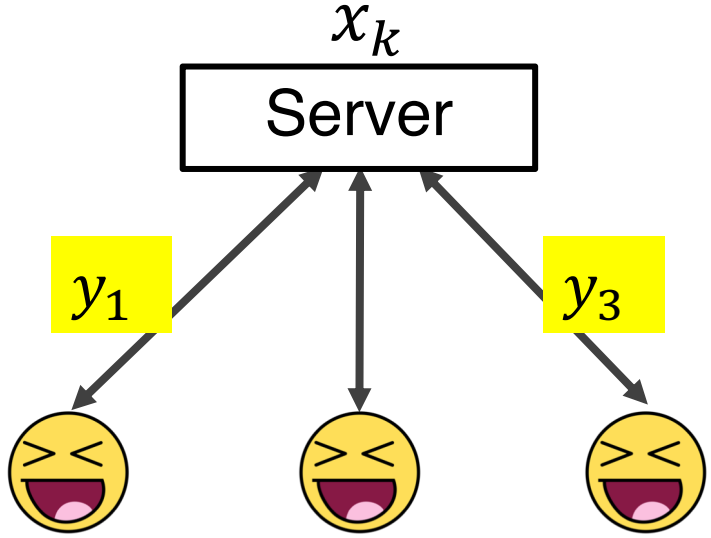
- Each agent $i$
  - Receive $x_k$ from server
  - Compute $y_k = x_k - \lambda_k \nabla f_i(x_k)$
  - Send $y_k$ to server

- Server updates estimate

$$x_{k+1} \leftarrow \frac{1}{n} \sum y_i$$

# Federated Architecture

- Server maintains estimate $x_k$




In each iteration

- Each agent $i$
  - Receive $x_k$ from server
  - Compute $y_k = x_k - \lambda_k \nabla f_i(x_k)$
  - Send $y_k$ to server

Replace single step by multiple steps

- Server updates estimate

$$x_{k+1} \leftarrow \frac{1}{n} \sum y_i$$

# Private Federated Learning [Bonawitz et al. 2016]

- Agents cooperate to compute $\sum y_i$ while preserving privacy of individual $y_i$ values

- Each pair of users communicates over secure channels to agree on matched pair of "noise"
  - User u sends noise $s_{u,v}$ to user v
  - User u obfuscates on $y_u$ value as below, for "large enough" R

$$z_u = ( y_u + \sum_v s_{v,u} - \sum_v s_{u,v} ) \mod R$$

  - Server can compute $\sum_u y_u$ correctly as $\sum_u z_u \mod R$ without being able to learn individual agent's estimate

Further improvement described in [Bonawitz et al. 2016]

# Secure Sum

- Similar concept introduced in [Abbe et al. 2012] for secure aggregation

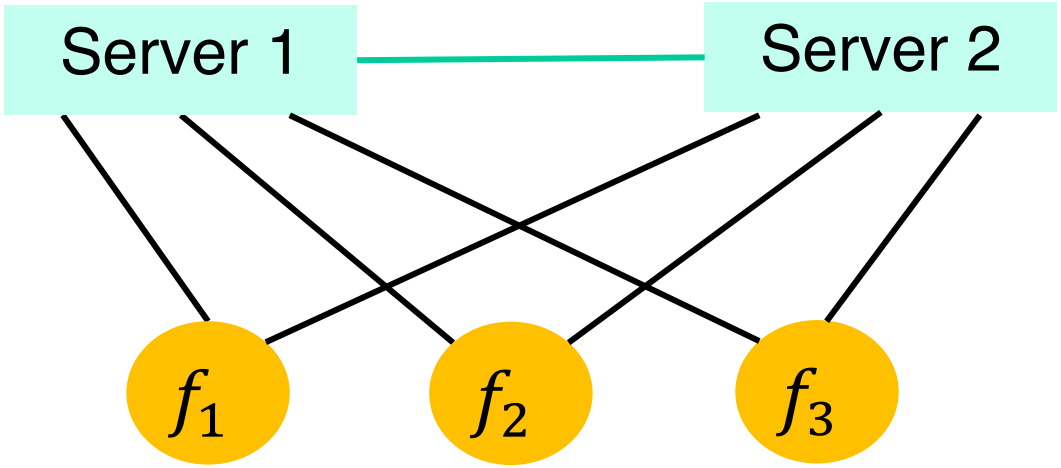
# Balanced Noise

- Exploit diversity … Multiple servers / neighbors
- Introduce noise that is “cancellable”

[Bonawitz et al. 2016, Abbe 2012] also introduce balanced noise, but it is utilized differently (also, no modulo operation used in schemes to be presented next)

# Server-Based Architecture: Balanced Noise

[Gade,Vaidya 2016, 2016a]



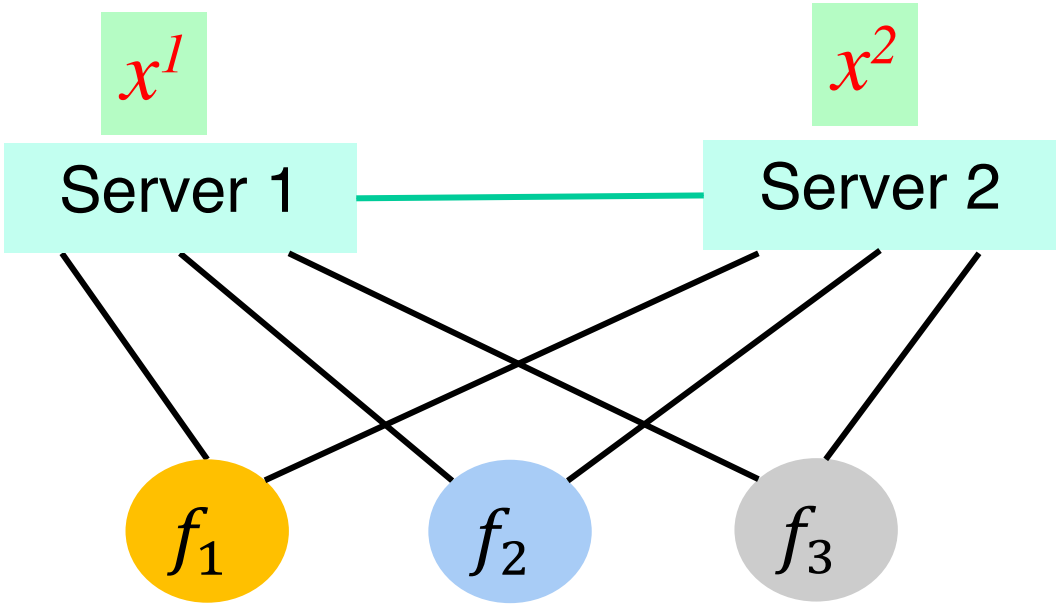

Privacy if subset of servers adversarial

# Balanced Noise

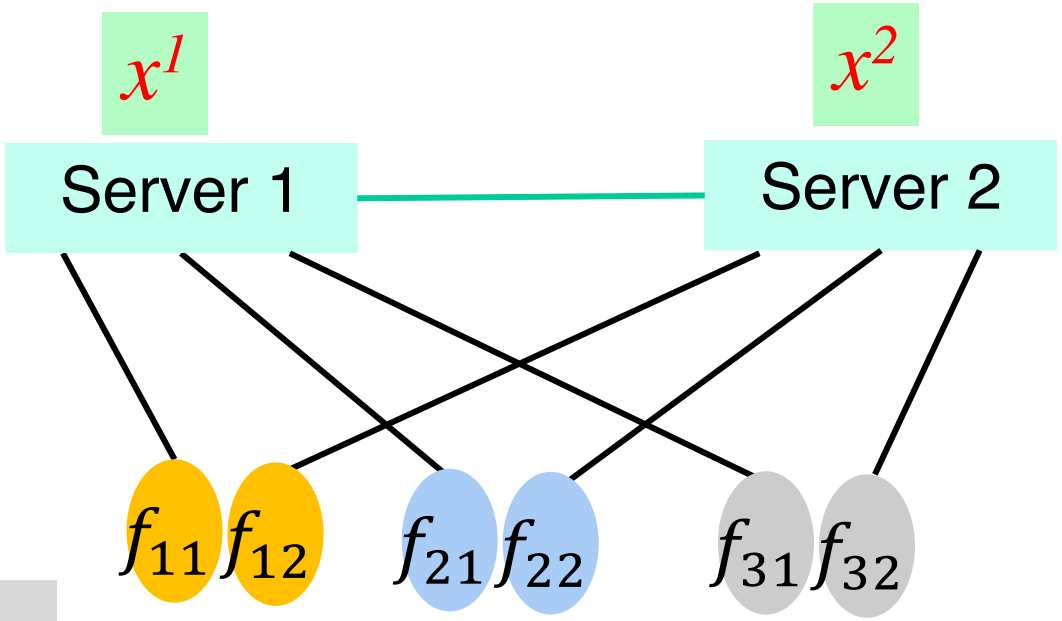
- Structured noise that

  *"cancels"* over servers/neighbors

# Intuition

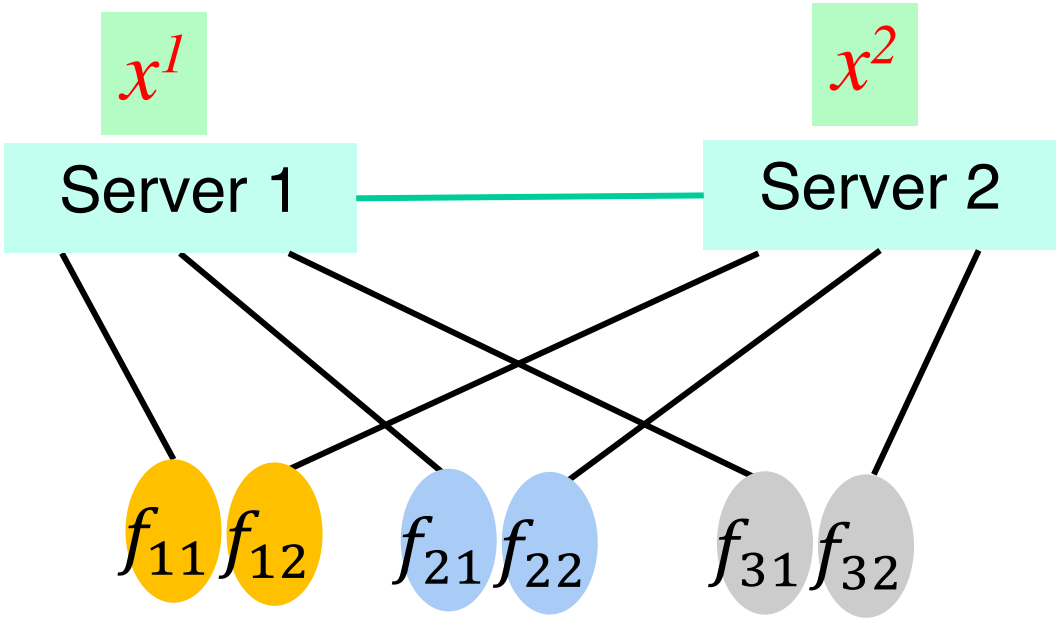


# Intuition




Each client simulates multiple clients

# Intuition




$f_{11}(x) + f_{12}(x) = f_1(x)$

$f_{ij}(x)$ not necessarily convex

# Algorithm

- Each server maintains an estimate

In each iteration

- Client $i$
  - Download estimates from corresponding server
  - Upload gradient of $f_i$
- Each server updates estimate using received gradients

# Algorithm

- Each server maintains an estimate

In each iteration

- Client $i$
  - Download estimates from corresponding server
  - Upload gradient of $f_i$
- Each server updates estimate using received gradients
- Servers periodically exchange estimates to perform a consensus step

# Claim

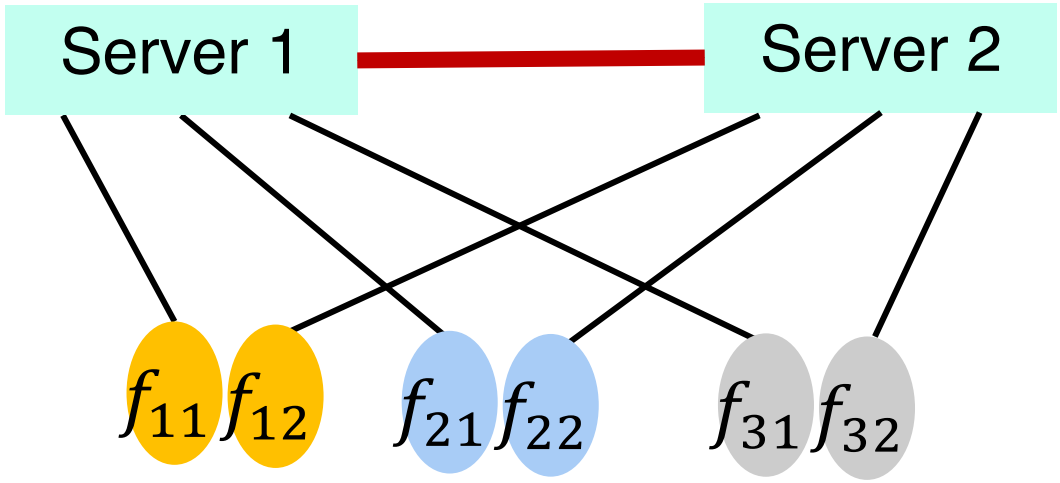
- Under suitable assumptions, servers eventually reach consensus in

$$argmin \sum_i f_i(x)$$

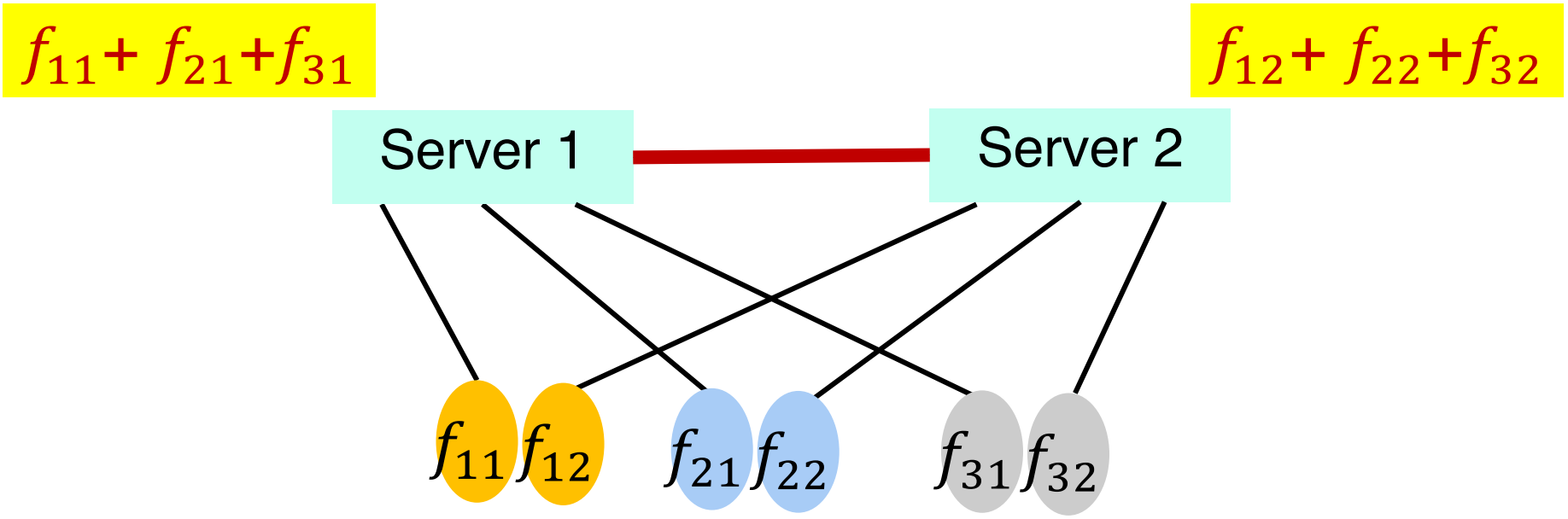
# Privacy

$f_{11}+ f_{21}+f_{31}$

$f_{12}+ f_{22}+f_{32}$



# Privacy




- Server 1 may learn $f_{11}$, $f_{21}$, $f_{31}$, $f_{12}+ f_{22}+f_{32}$
- Not sufficient to learn $f_i$

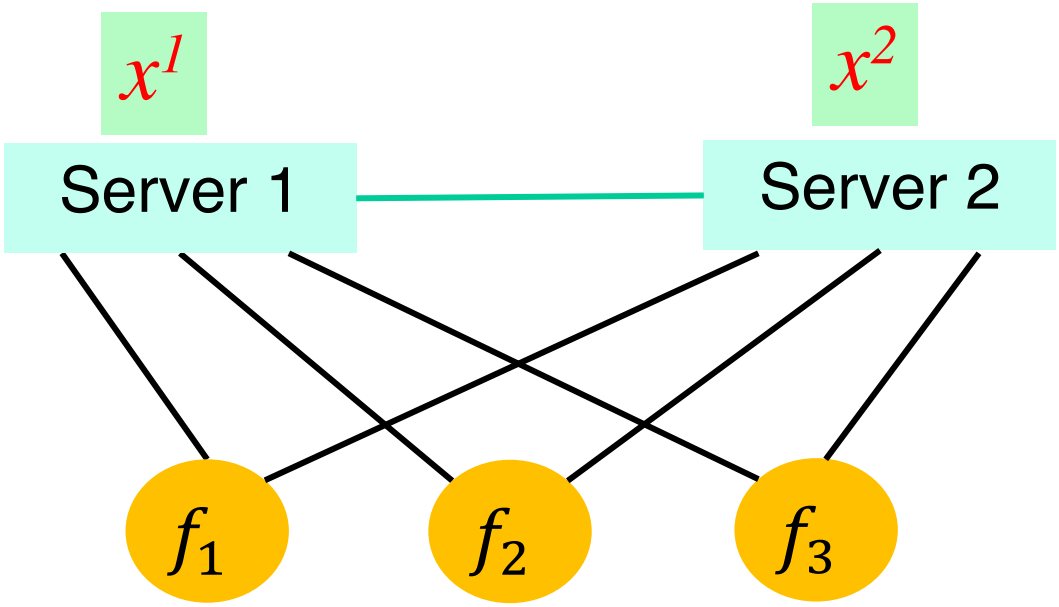
$$f_{11}(x) + f_{12}(x) = f_1(x)$$

- *Function splitting* not necessarily practical
- Structured randomization as an alternative

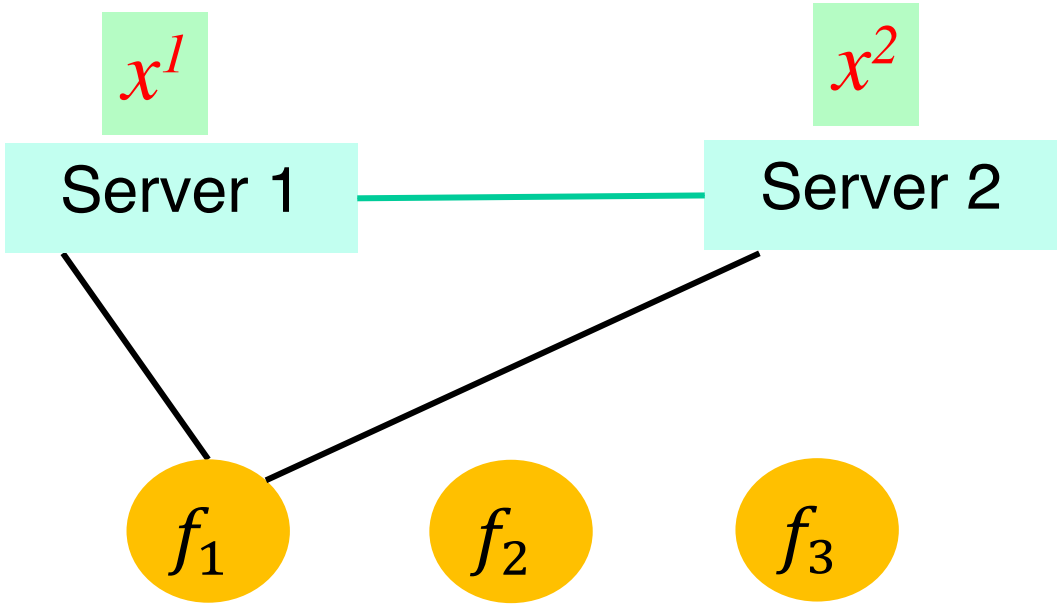
# Structured Randomization

- Multiplicative or additive noise in gradients
- Noise *cancels* over servers

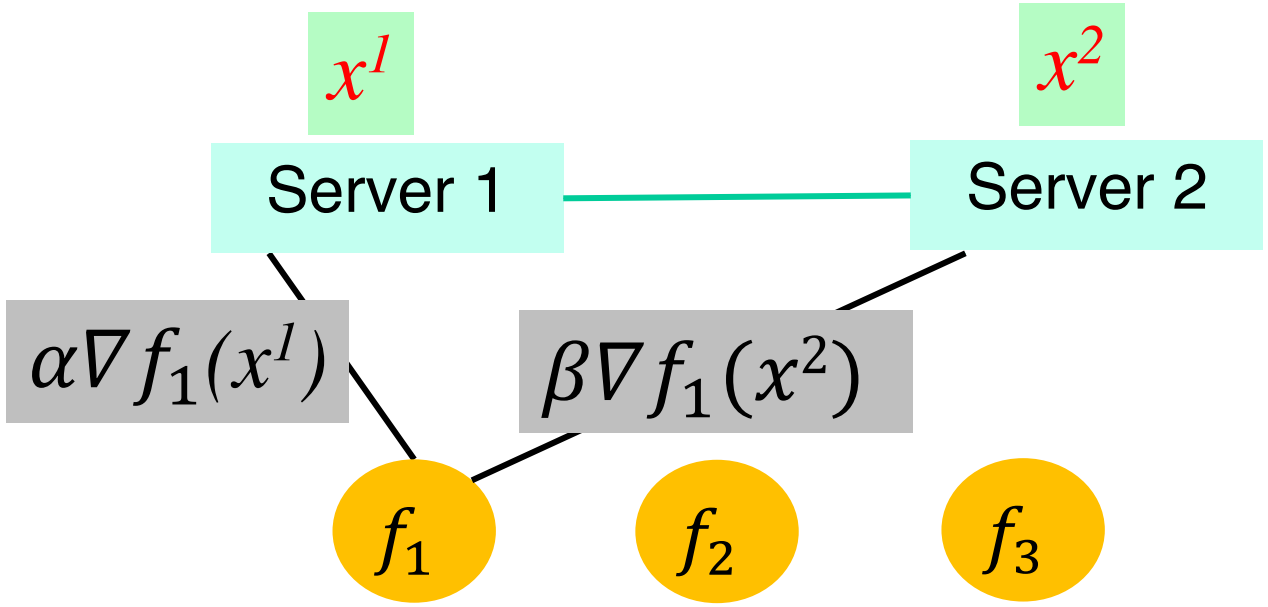
# Multiplicative Noise



# Multiplicative Noise

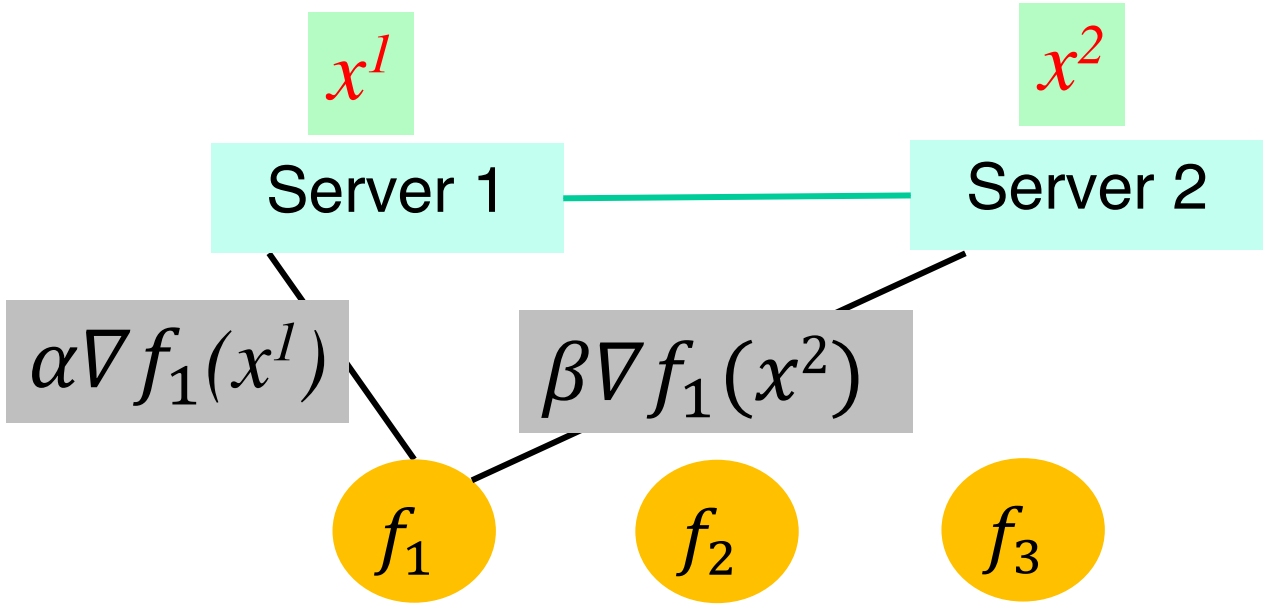


# Multiplicative Noise




$\alpha+\beta=1$

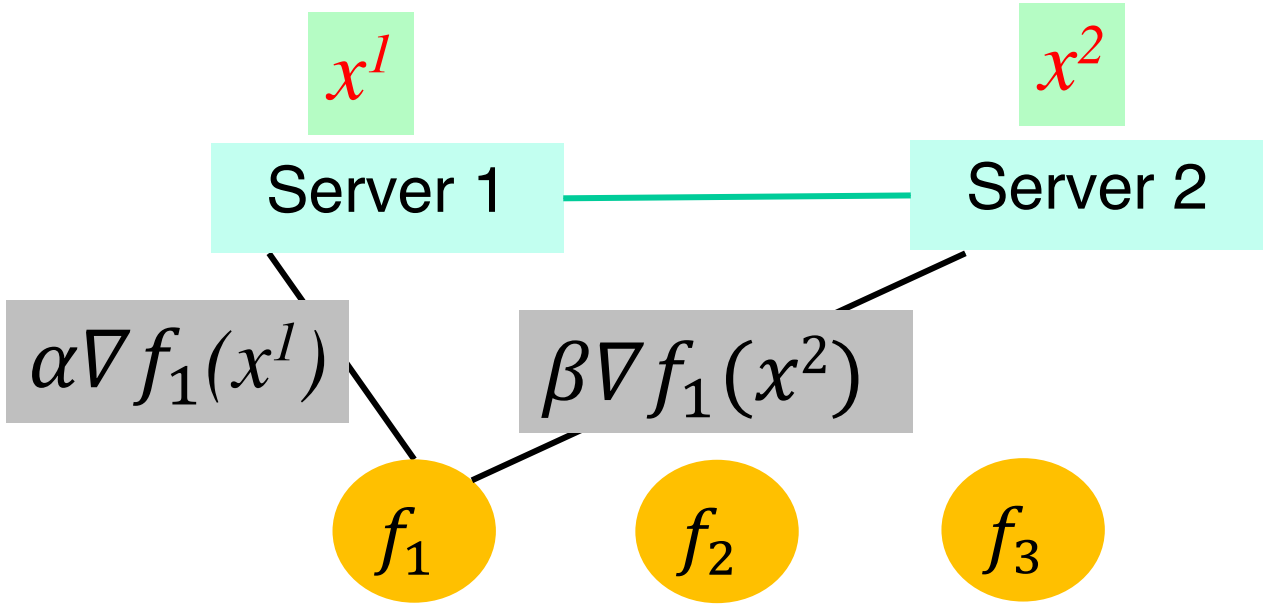
# Multiplicative Noise




$\alpha+\beta=1$

Suffices for this invariant to hold over a larger number of iterations

# Multiplicative Noise




$\alpha+\beta=1$

Noise from client *i* to server *j* *not* zero-mean

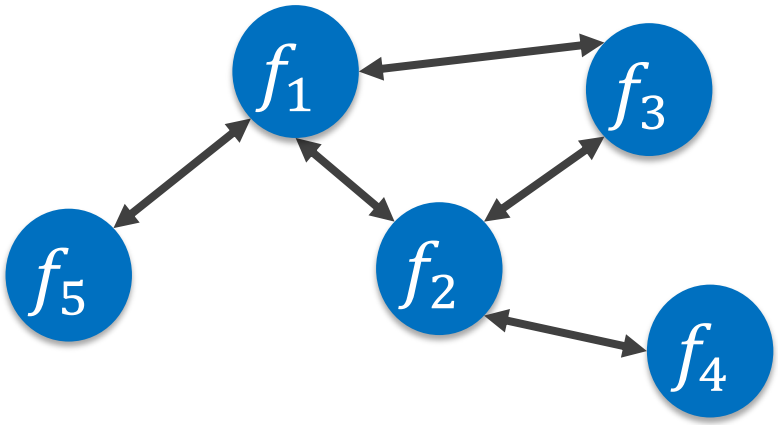
# Claim

- Under suitable assumptions, servers eventually reach consensus in

$$argmin \sum_i f_i(x)$$

# Peer-to-Peer Architecture

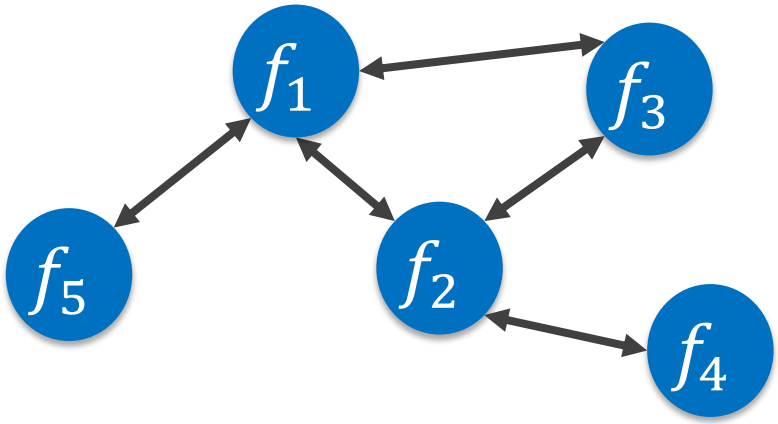
[Gade, Vaidya 2016, 2017, Gupta,Chopra 2019, Gupta et al. 2020]



# Balanced Noise

[Gade, Vaidya 2016, 2017]

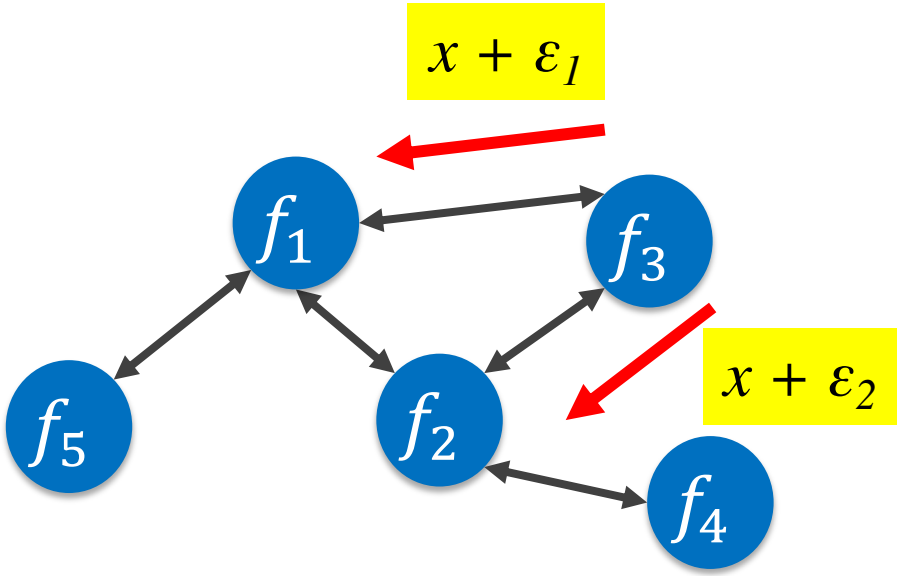
- Each agent shares noisy estimate with neighbors
  - Scheme 1 – Noise cancels over neighbors
  - Scheme 2 – Noise cancels network-wide



# Balanced Noise

[Gade, Vaidya 2016, 2017]

- Each agent shares noisy estimate with neighbors
  - Scheme 1 – Noise cancels over neighbors
  - Scheme 2 – Noise cancels network-wide




$\varepsilon_1 + \varepsilon_2 = 0$ (over iterations)

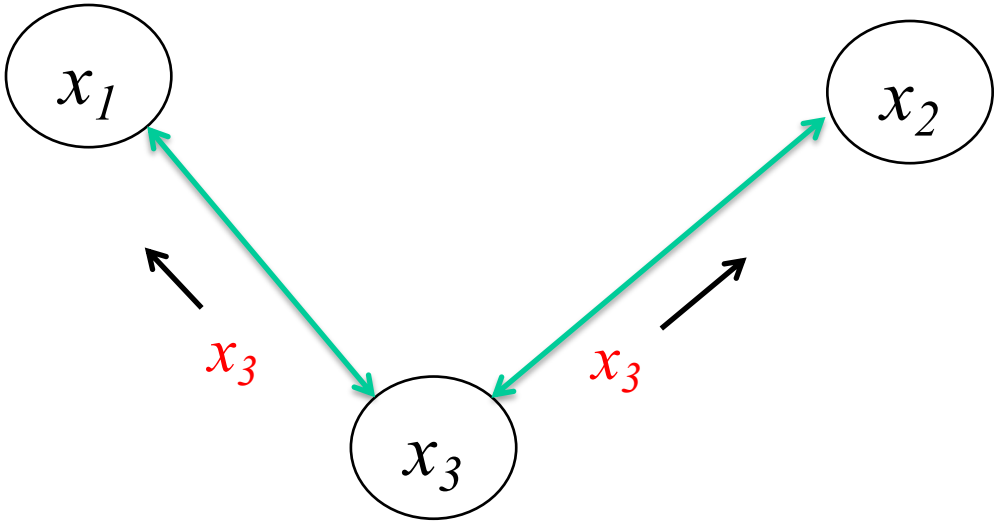
# Locally Balanced Noise

Perturbations

- Add to zero (locally per node)

Algorithm

- Node j selects $\mathrm{d}_{\mathrm{k}}^{\mathrm{j,i}}$ such that $\sum_{\mathrm{i}} \mathrm{d}_{\mathrm{k}}^{\mathrm{j,i}} = 0$ and $\left|d_k^{j,i}\right| \leq \Delta$
- Share $\mathrm{w}_{\mathrm{k}}^{\mathrm{j,i}} = \mathrm{x}_{\mathrm{k}}^{\mathrm{j}} + \mathrm{d}_{\mathrm{k}}^{\mathrm{j,i}}$ with node I
- Weighted averaging and gradient descent




$$x_1[t+1] = \frac{2}{3}x_1[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_1(x_1[t])$$

$$x_3[t+1] = \frac{1}{3}x_1[t] + \frac{1}{3}x_2[t] + \frac{1}{3}x_3[t] - \lambda_t \nabla f_3(x_3[t])$$

# Network Balanced Noise

Perturbations

- Add to zero (over network)

Algorithm

- Node j computes perturbation $d_k^j$
  - sends $s^{j,i}$ to i
  - add received $s^{i,j}$ and subtract sent $s^{j,i} \Rightarrow d_k^j = \sum \text{rcvd} - \sum \text{sent}$
- Obfuscate state $w_k^j = x_k^j + d_k^j$ shared with neighbors
- Weighted averaging and gradient descent

# Function Sharing

- Let $f_i(x)$ be bounded degree polynomials

Algorithm

- Node j shares $s^{j,i}(x)$ with node i
- Node j obfuscates using $p_j(x) = \sum s^{i,j}(x) - \sum s^{j,i}(x)$
- Use $\hat{f}_j(x) = f_j(x) + p_j(x)$ and use distributed gradient descent

# Function Sharing

- Function Sharing iterates converge to correct optimum ($\sum \hat{f}_i(x) = f(x)$)

- Privacy: If vertex connectivity of graph $\geq$ f+1 then no group of f nodes can estimate function $f_i$ (or sum of functions of any subset of functions)

# Summary

- Several different approaches for privacy in optimization/learning
- More work needed to develop practical schemes that maintain accuracy and yet achieve strong privacy guarantees

# Summary

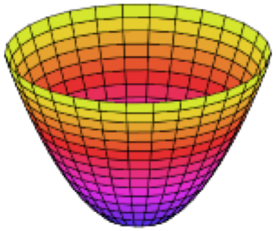
- Tutorial provides only a limited overview of past work
- Longer version of the tutorial available from http://disc.georgetown.domains/talks.htm

# Acknowledgements

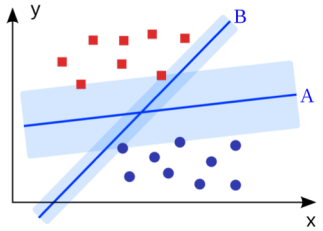

- Some of our results presented in the tutorial resulted from work supported in part by the National Science Foundation and Army Research Laboratory. The views and conclusions presented are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory, National Science Foundation or the U.S. Government.

# Attribution for Images

- Krishnavedala, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons
  https://upload.wikimedia.org/wikipedia/commons/1/12/Paraboloid_of_Revolution.svg
  https://commons.wikimedia.org/wiki/File:Paraboloid_of_Revolution.svg



- Ennepetaler86, CC BY 3.0 <https://creativecommons.org/licenses/by/3.0>, via Wikimedia Commons
  https://upload.wikimedia.org/wikipedia/commons/f/f2/Svm_intro.svg
  https://commons.wikimedia.org/wiki/File:Svm_intro.svg




- Zufzzi, Public domain, via Wikimedia Commons
  https://commons.wikimedia.org/wiki/File:Neural_network_bottleneck_achitecture.svg
  https://upload.wikimedia.org/wikipedia/commons/8/8b/Neural_network_bottleneck_achitecture.svg