LightSecAgg: a Lightweight and Versatile Design for Secure Aggregation in Federated Learning

Jinhyun So

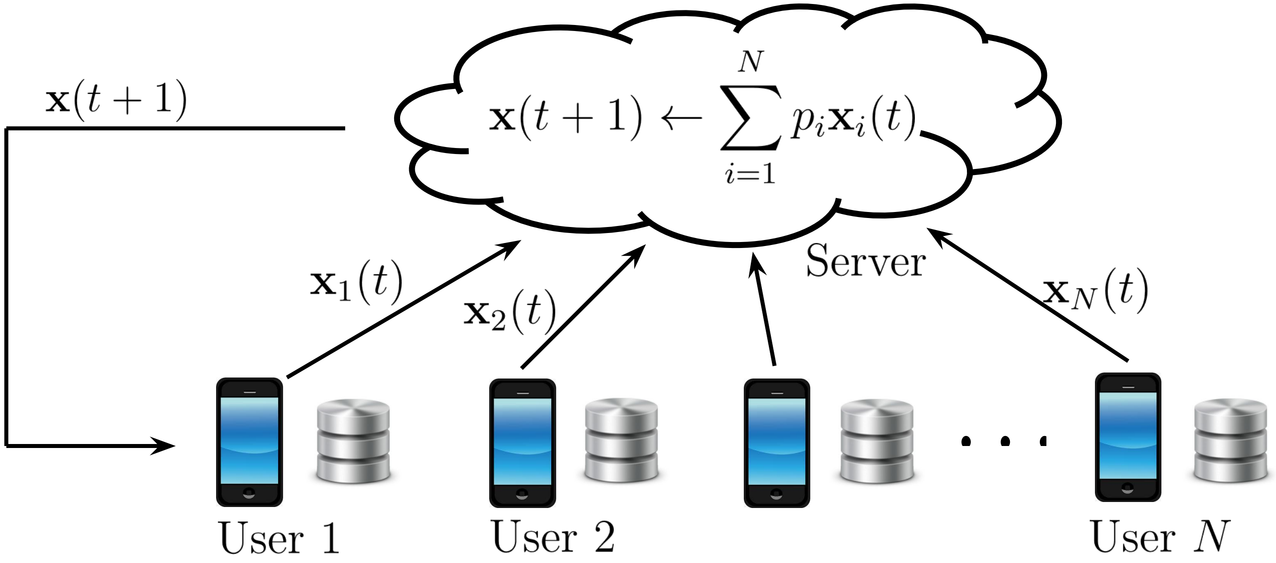
University of Southern California

Joint work with

Chaoyang He(FedML), Chien-Sheng Yang (MediaTek), Songze Li (HKUST), Qian Yu (Princeton), Ramy E. Ali (Samsung), Basak Guler (UCR), and Salman Avestimehr (USC, FedML)

Federated Learning

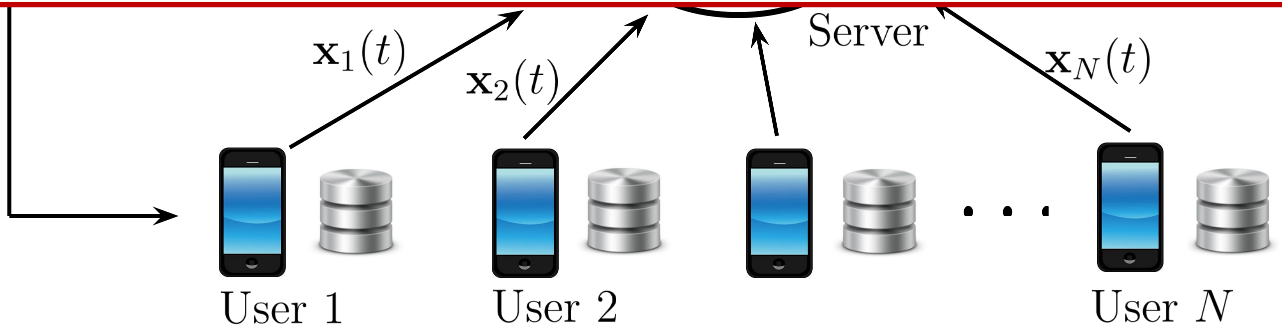
Machine learning on massive amount of data collected on many users/mobile devices



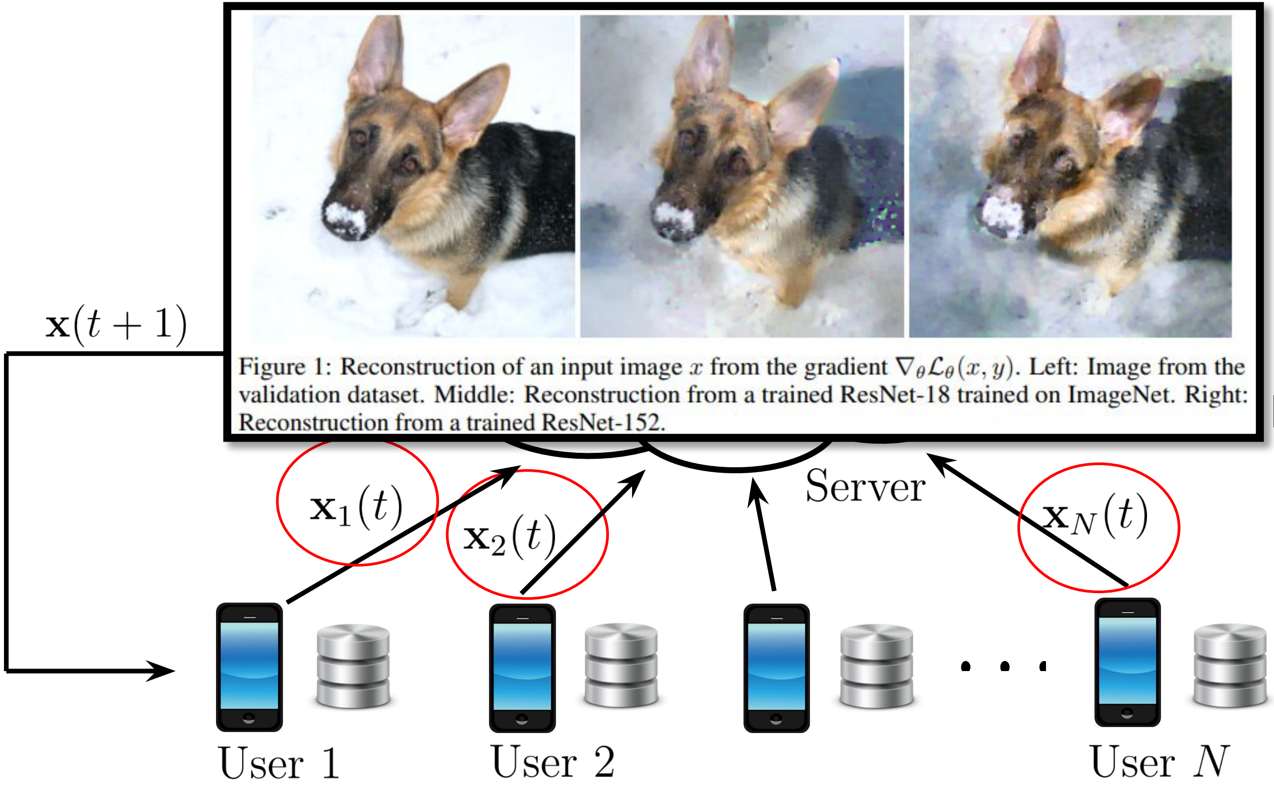
Federated Learning

Machine learning on massive amount of data collected on many users/mobile devices

- Key Design Principles
  1. Privacy
  2. System Efficiency



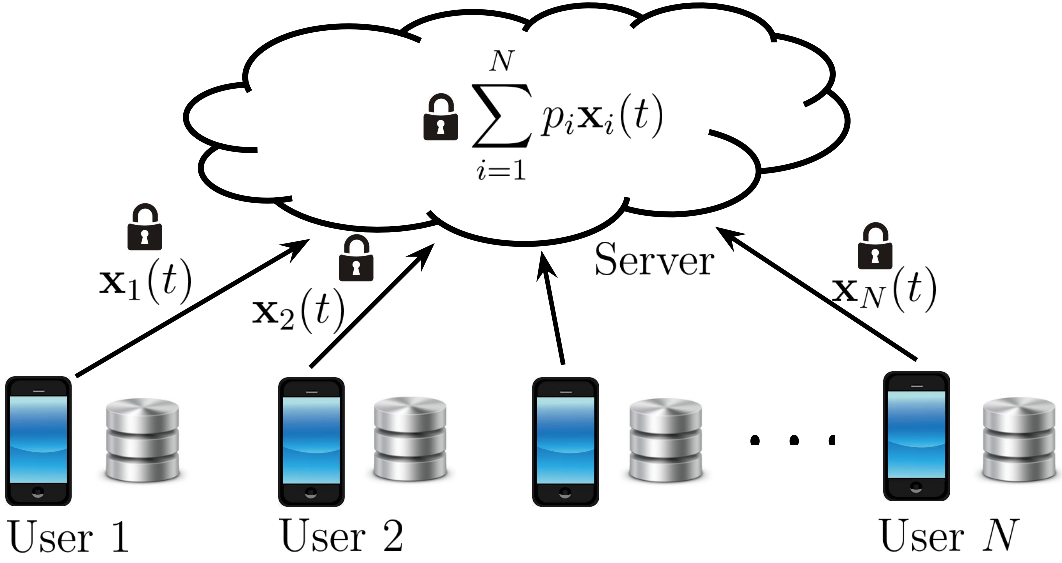
Model Inversion Attack

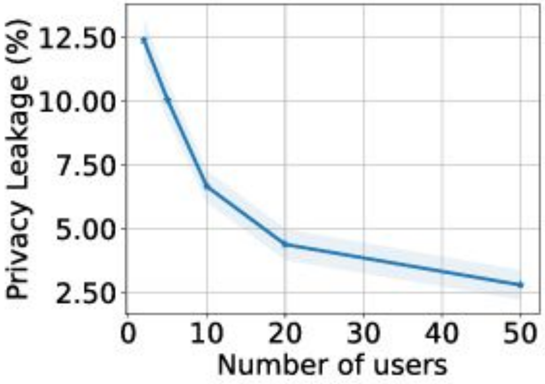


[Geiping et al]

Problem: Individual model update can leak sensitive data.

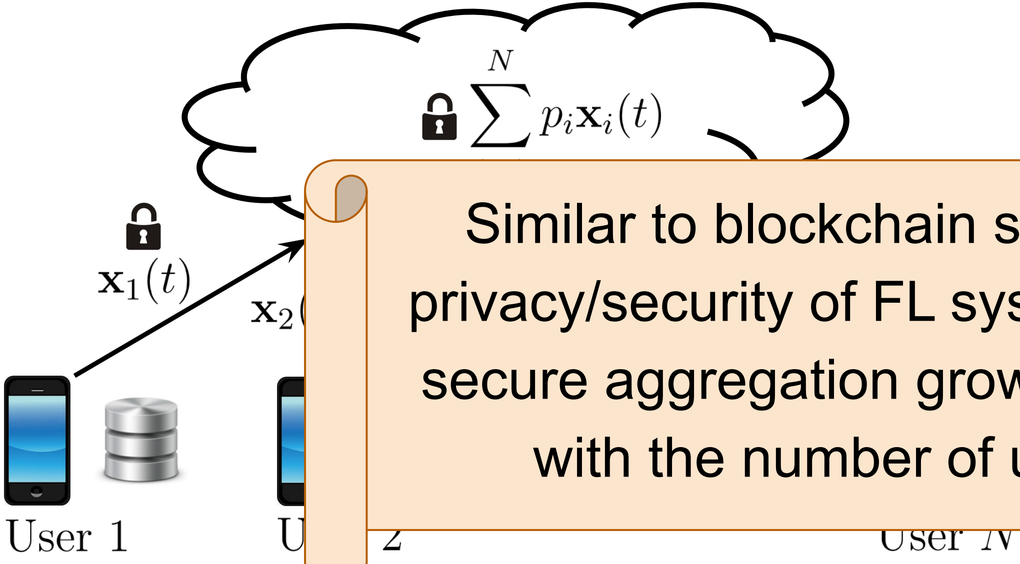
Key component: Secure Aggregation

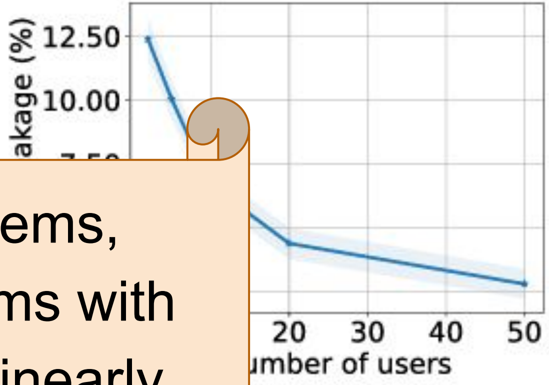




Elkordy, A. R., Zhang, J., Ezzeldin, Y. H., Psounis, K., & Avestimehr, S. (2022). **How Much Privacy Does Federated Learning with Secure Aggregation Guarantee?**. *arXiv preprint arXiv:2208.02304*.

Key component: Secure Aggregation





Similar to blockchain systems, privacy/security of FL systems with secure aggregation grows linearly with the number of users

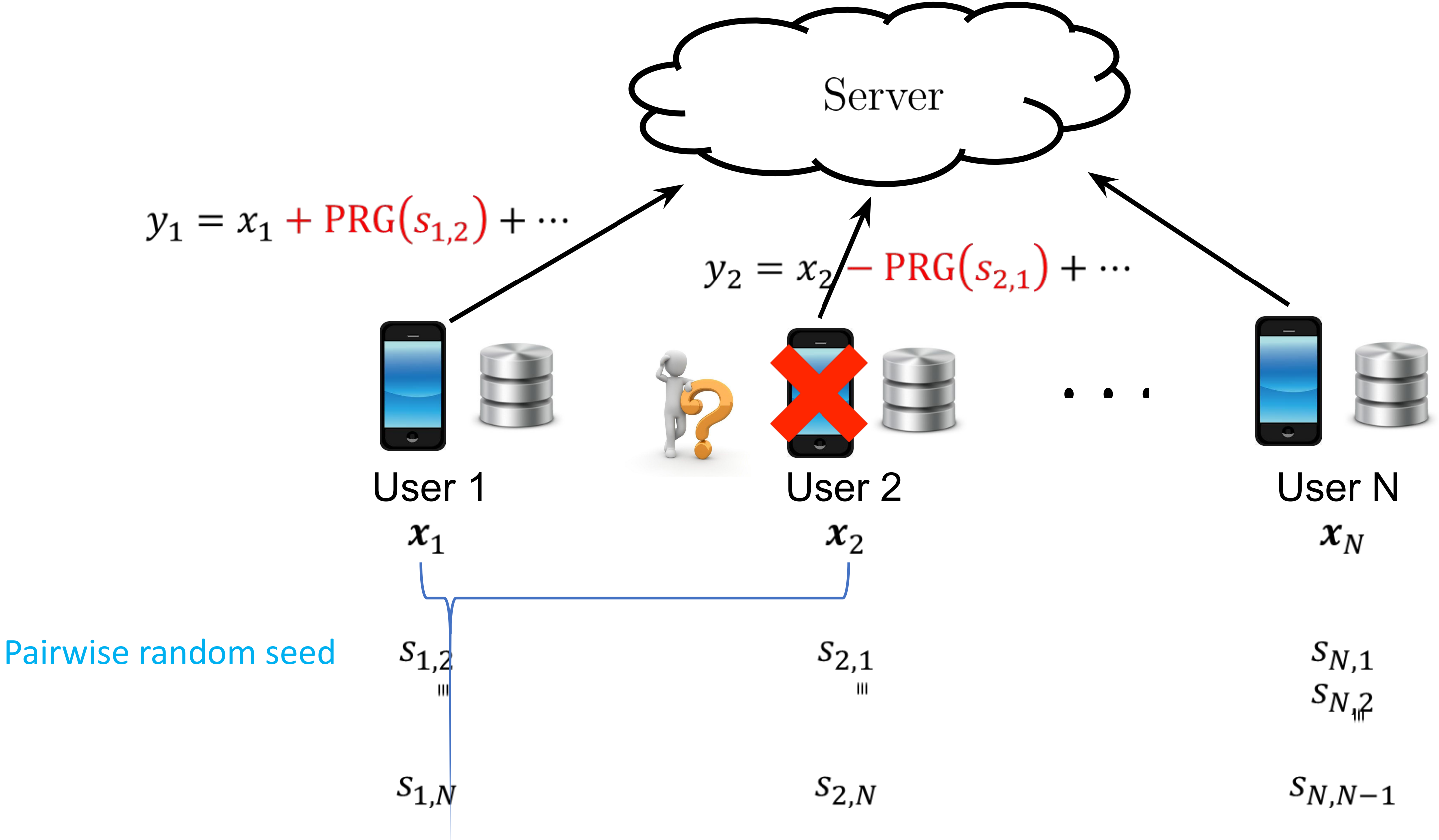
Elkordy, A. R., Zhang, J., Ezzeldin, Y. H., Psounis, K., & Avestimehr, S. (2022). **How Much Privacy Does Federated Learning with Secure Aggregation Guarantee?**. *arXiv preprint arXiv:2208.02304*.

Key component: Secure Aggregation

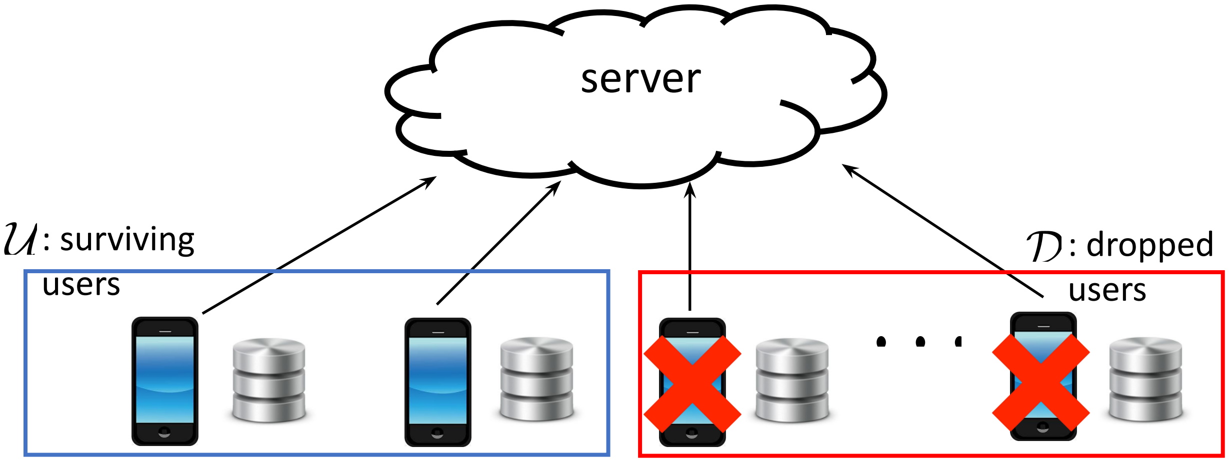
Practical Secure Aggregation for Privacy-Preserving Machine Learning

Keith Bonawitz[*], Vladimir Ivanov[*], Ben Kreuter[*],
Antonio Marcedone[†‡],H. Brendan McMahan[*], Sarvar Patel[*],
Daniel Ramage[*], Aaron Segal[*], and Karn Seth[*]
[*]{bonawitz,vlivan,benkreuter,mcmahan,
sarvar,dramage,asegal,karn}@google.com
Google, Mountain View, CA 94043
[‡]marcedone@cs.cornell.edu
Cornell Tech, 2 West Loop Rd., New York, NY 10044

State-of-the-Art: SecAgg



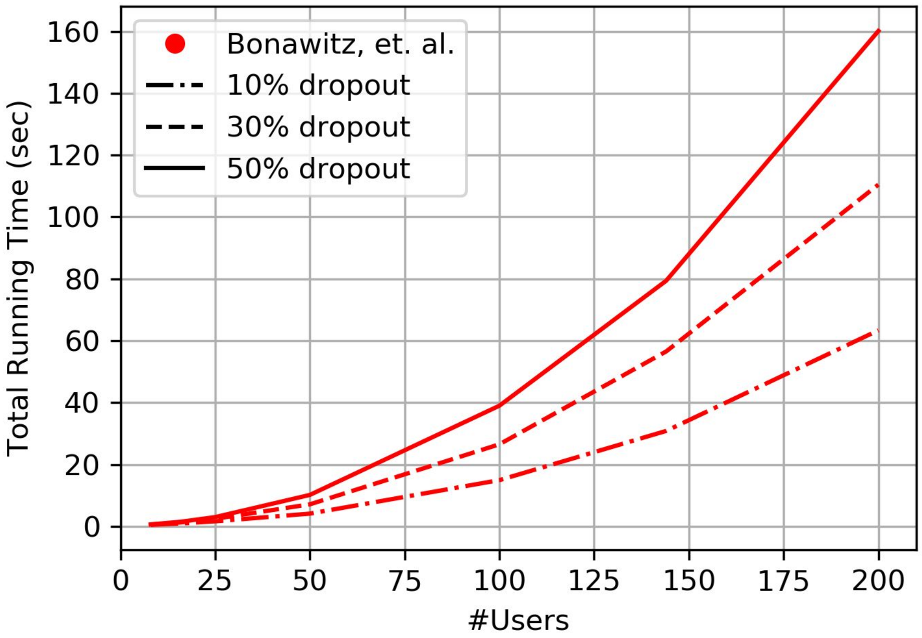
State-of-the-Art: SecAgg



$$\mathbf{z} = \sum_{u \in \mathcal{U}} (\mathbf{y}_u - \mathrm{PRG}(b_u)) + \sum_{u \in \mathcal{D}} \left(\sum_{v: u<v} \mathrm{PRG}(s_{u,v}) - \sum_{v: u>v} \mathrm{PRG}(s_{v,u}) \right)$$

The number of mask reconstructions at the server substantially grows as more users are dropped, causing a major computational bottleneck.

State-of-the-Art: SecAgg



Individual model size of 100,000 with 32 bits entries- experiments over Amazon EC2

Key component: Secure Aggregation

- Aggregation complexity is the **MAIN BOTTLENECK.**
- Some works reduce the complexity, but sacrifice the dropout/privacy guarantees.

	Complexity	**Privacy/Dropout Guarantee**
SecAgg [Bonawitz, 17’]	$O(N^2)$	Strong (worst-case)
SecAgg+ [Bell, 20’]	$O(N \log N)$	Weak (average-case)
Turbo-Aggregate [So, 21’]	$O(N \log N)$	Weak (average-case)
FastSecAgg [Kadhe, 21’]	$O(N \log N)$	Weak (average-case)

New Perspective

State-of-the-art:

$$\mathbf{z} = \sum_{u\in\mathcal{U}} (\mathbf{y}_u - \mathrm{PRG}(b_u)) + \sum_{u\in\mathcal{D}} \left(\sum_{v:u<v} \mathrm{PRG}(s_{u,v}) - \sum_{v:u>v} \mathrm{PRG}(s_{v,u}) \right)$$

New Perspective:

$$\mathbf{z} = \sum_{u\in\mathcal{U}} \mathbf{x}_u + \sum_{u\in\mathcal{U}} \mathbf{z}_u$$

Our focus

We turn the focus from "random-seed reconstruction of the dropped users" to "one-shot aggregate-mask reconstruction of the surviving users".

New Perspective

$\sum_{i\in\mathcal{S}} z_i$?

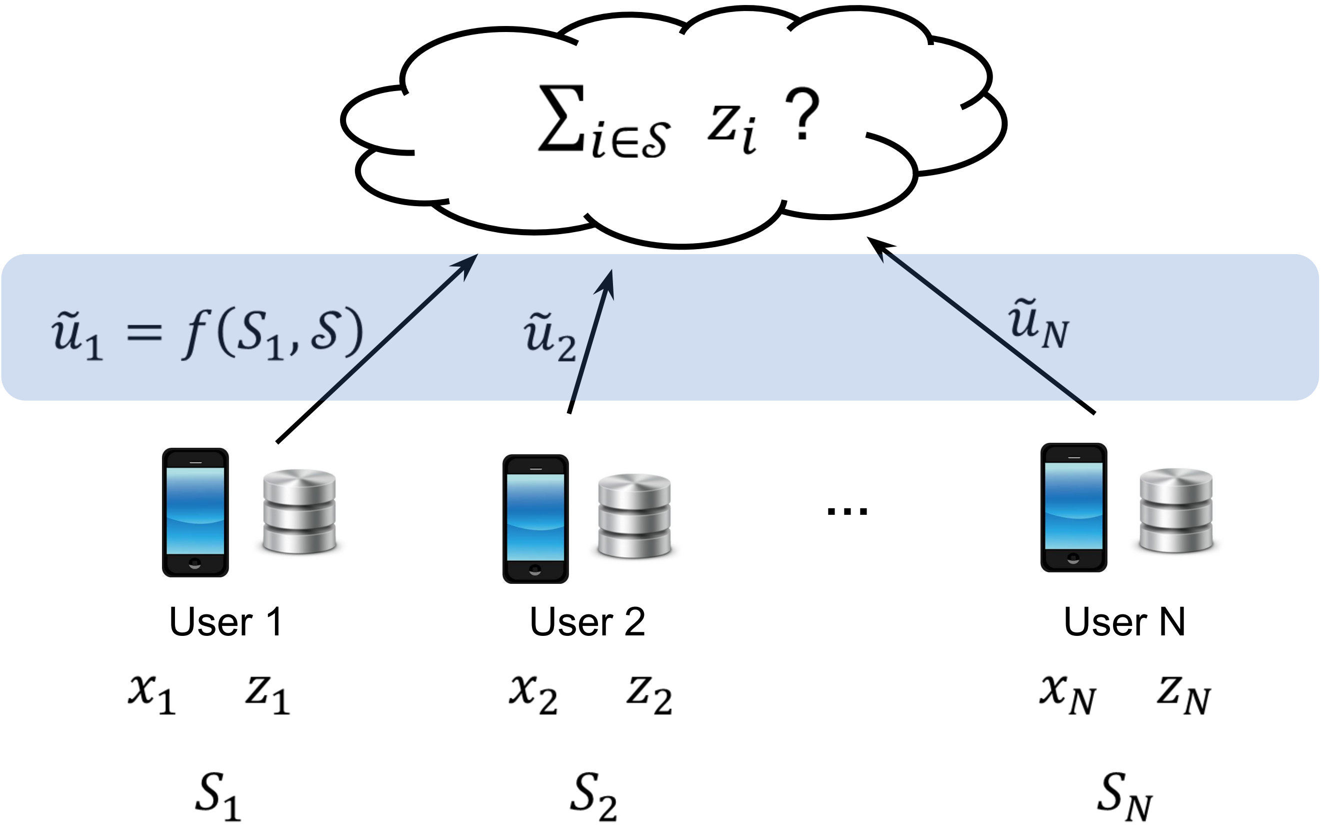
User 1   User 2   ...   User N

$x_1 \quad z_1 \qquad x_2 \quad z_2 \qquad x_N \quad z_N$

$$S_1 = \begin{bmatrix} s_{1,1} \\ s_{2,1} \\ \vdots \\ s_{N,1} \end{bmatrix} \qquad S_2 = \begin{bmatrix} s_{1,2} \\ s_{2,2} \\ \vdots \\ s_{N,2} \end{bmatrix} \qquad S_N = \begin{bmatrix} s_{1,N} \\ s_{2,N} \\ \vdots \\ s_{N,N} \end{bmatrix}$$

Step 1.
Encoding
& Secret Sharing

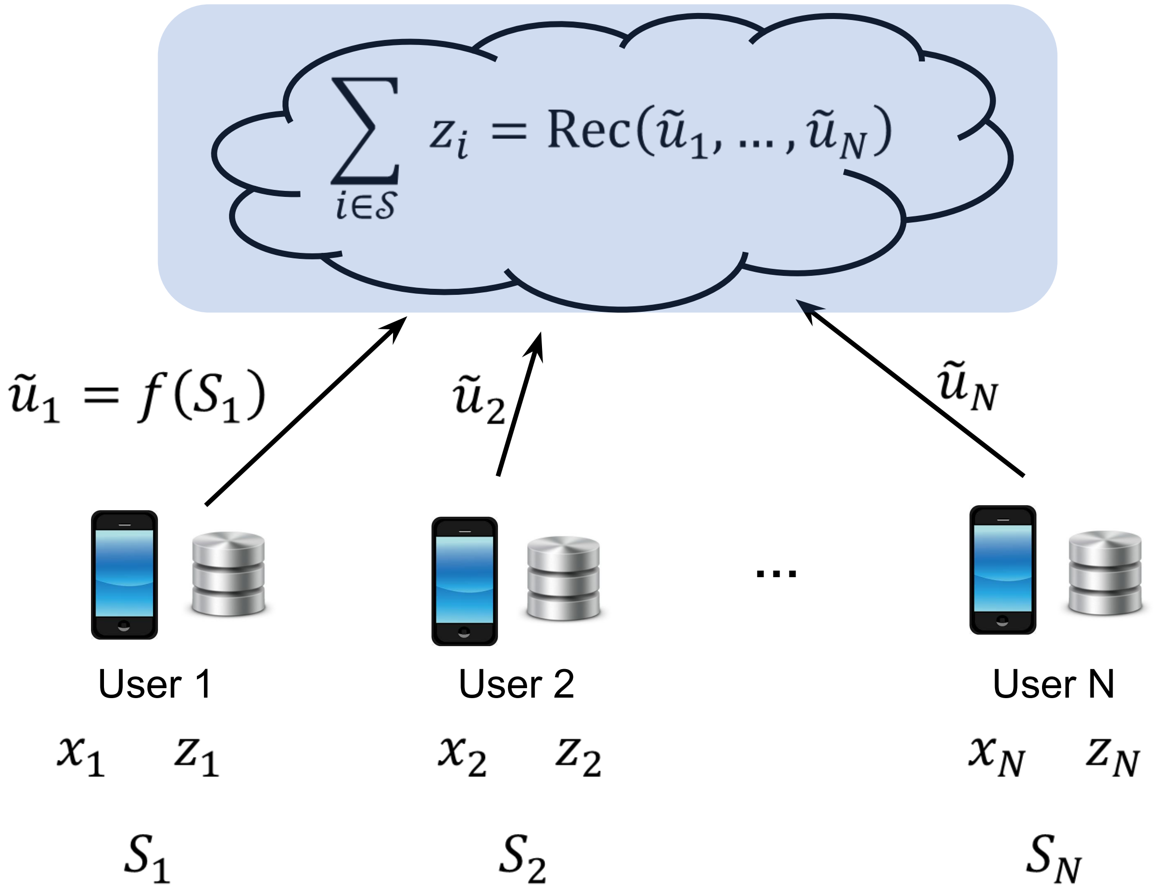
New Perspective



Step 2.
Uploading shares

Step 1.
Encoding
& Secret Sharing

New Perspective

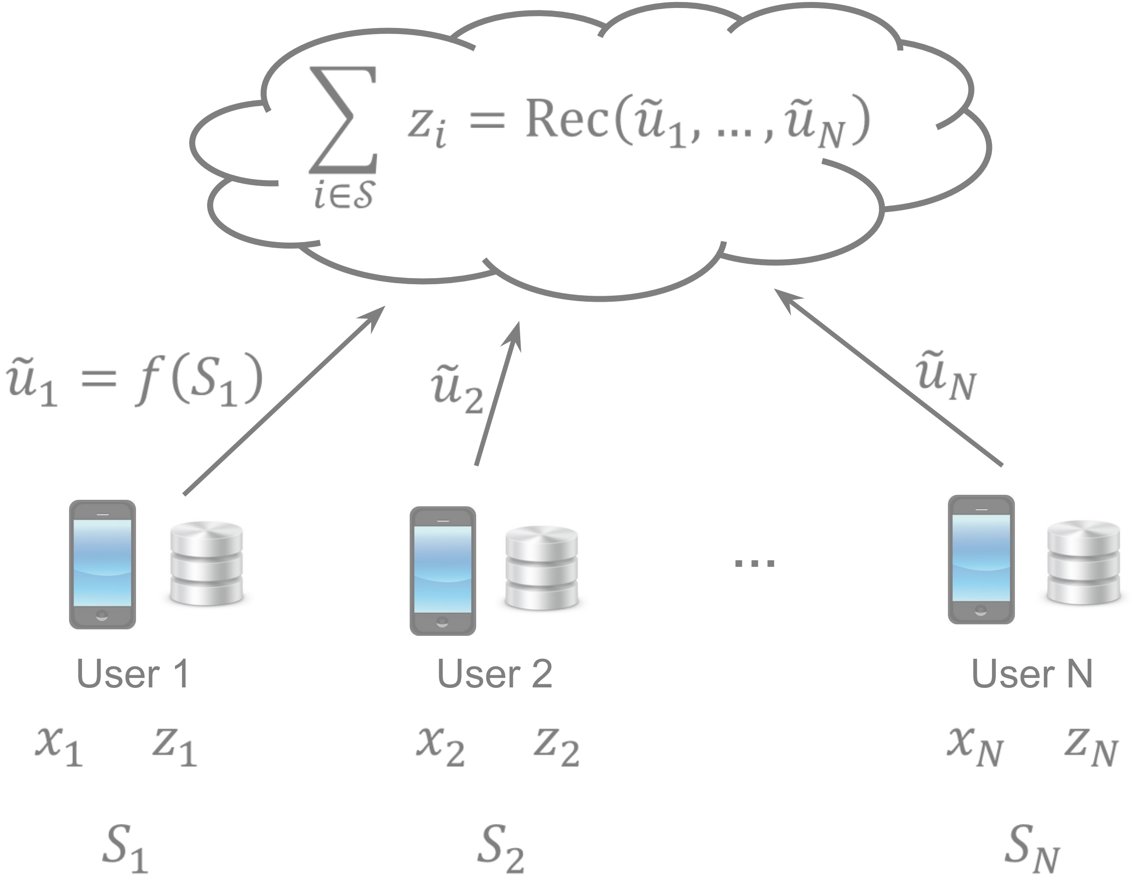


Step 3.
Reconstruction

Step 2.
Uploading shares

Step 1.
Encoding
& Secret Sharing

New Perspective



Three Objectives

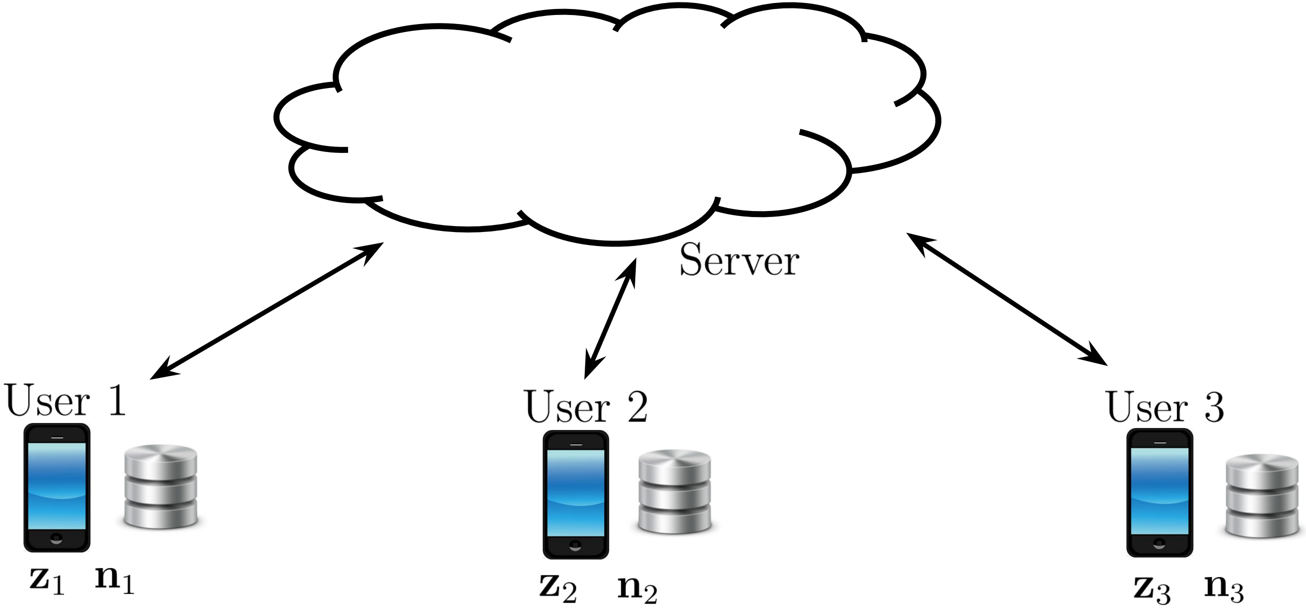
1) Reconstruction of $\sum_{i \in \mathcal{S}} z_i$ for any $\mathcal{S}$

2) Compactness in comm. & comp.

3) Privacy of z_i's

Example (LightSecAgg)

1) Offline encoding and sharing of local masks



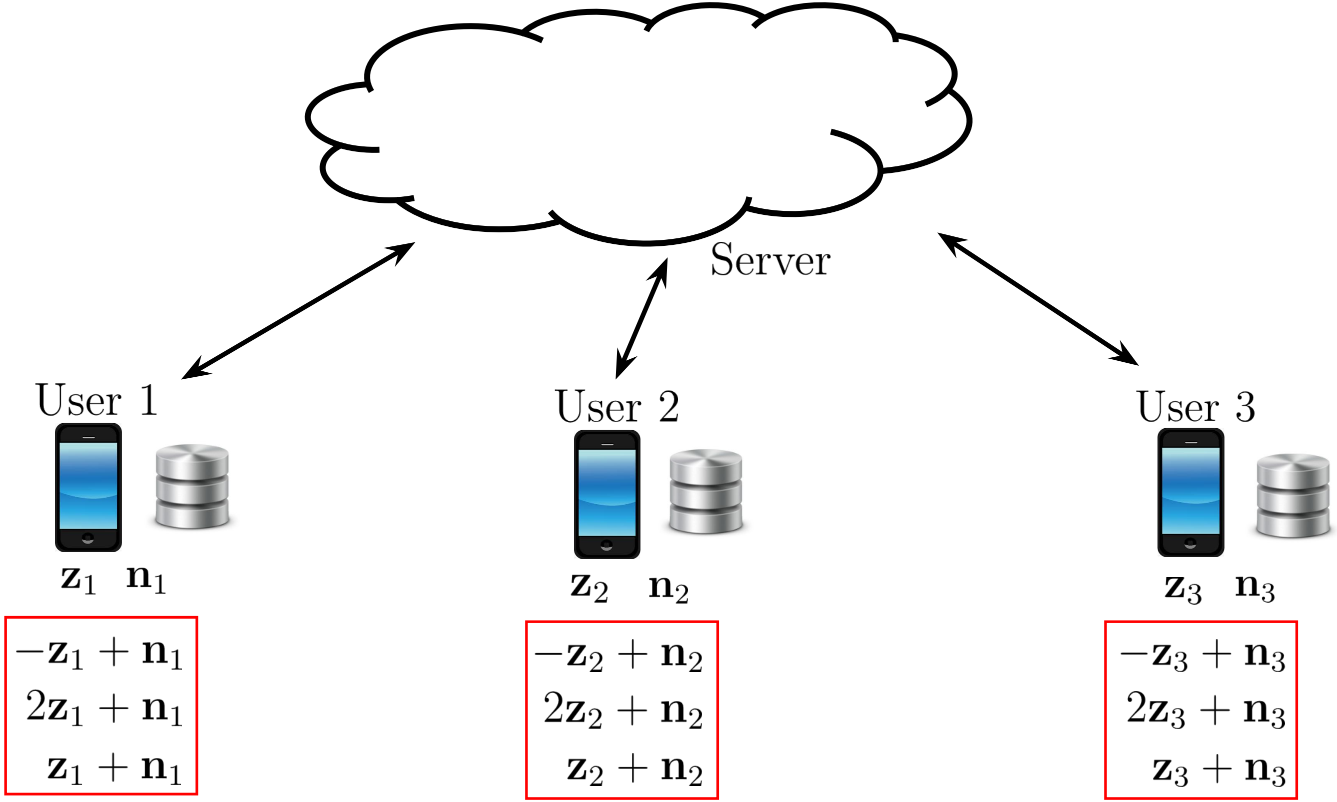
Encoding via MDS

$$\boxed{\begin{array}{r} -\mathbf{z}_1 + \mathbf{n}_1 \\ 2\mathbf{z}_1 + \mathbf{n}_1 \\ \mathbf{z}_1 + \mathbf{n}_1 \end{array}}$$

1) $\mathbf{z}_1$ and $\mathbf{n}_1$ can be recovered by any 2 out of 3 encoded masks.
   => robustness against dropped users
2) $\mathbf{n}_1$ protects $\mathbf{z}_1$ => privacy against the other users

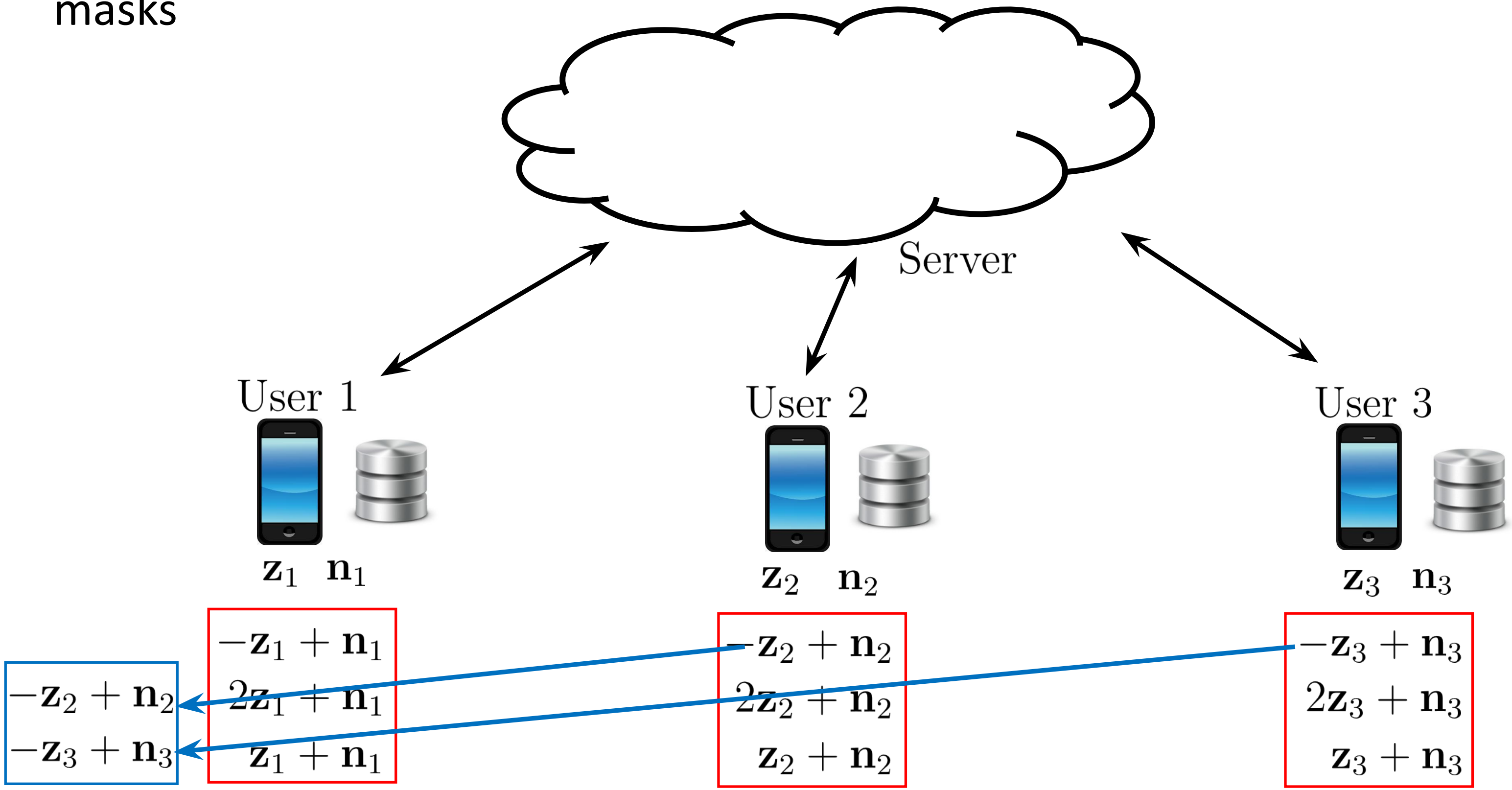
Example (LightSecAgg)

1) Offline encoding and sharing of local masks



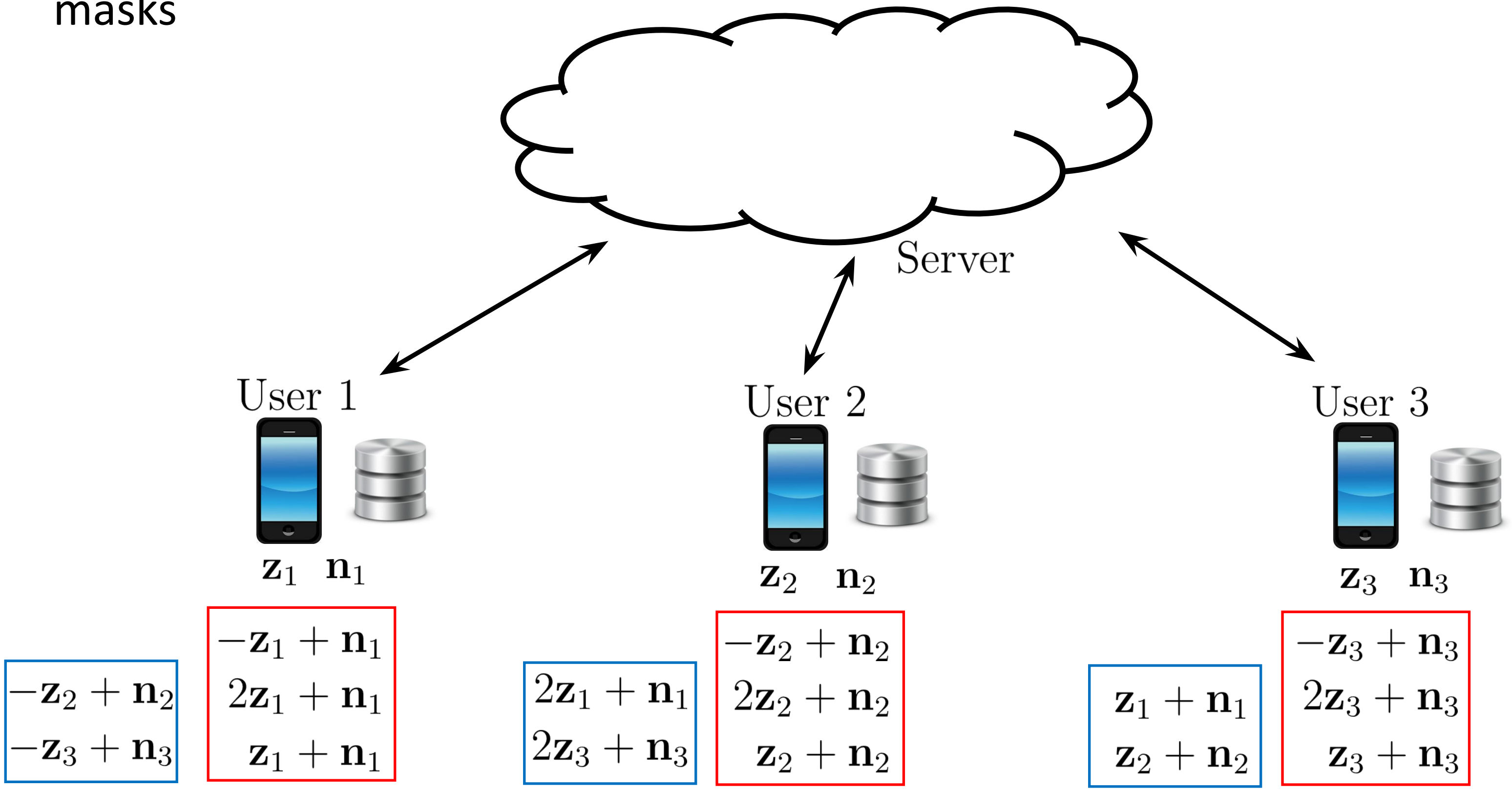
Example (LightSecAgg)

1) Offline encoding and sharing of local masks



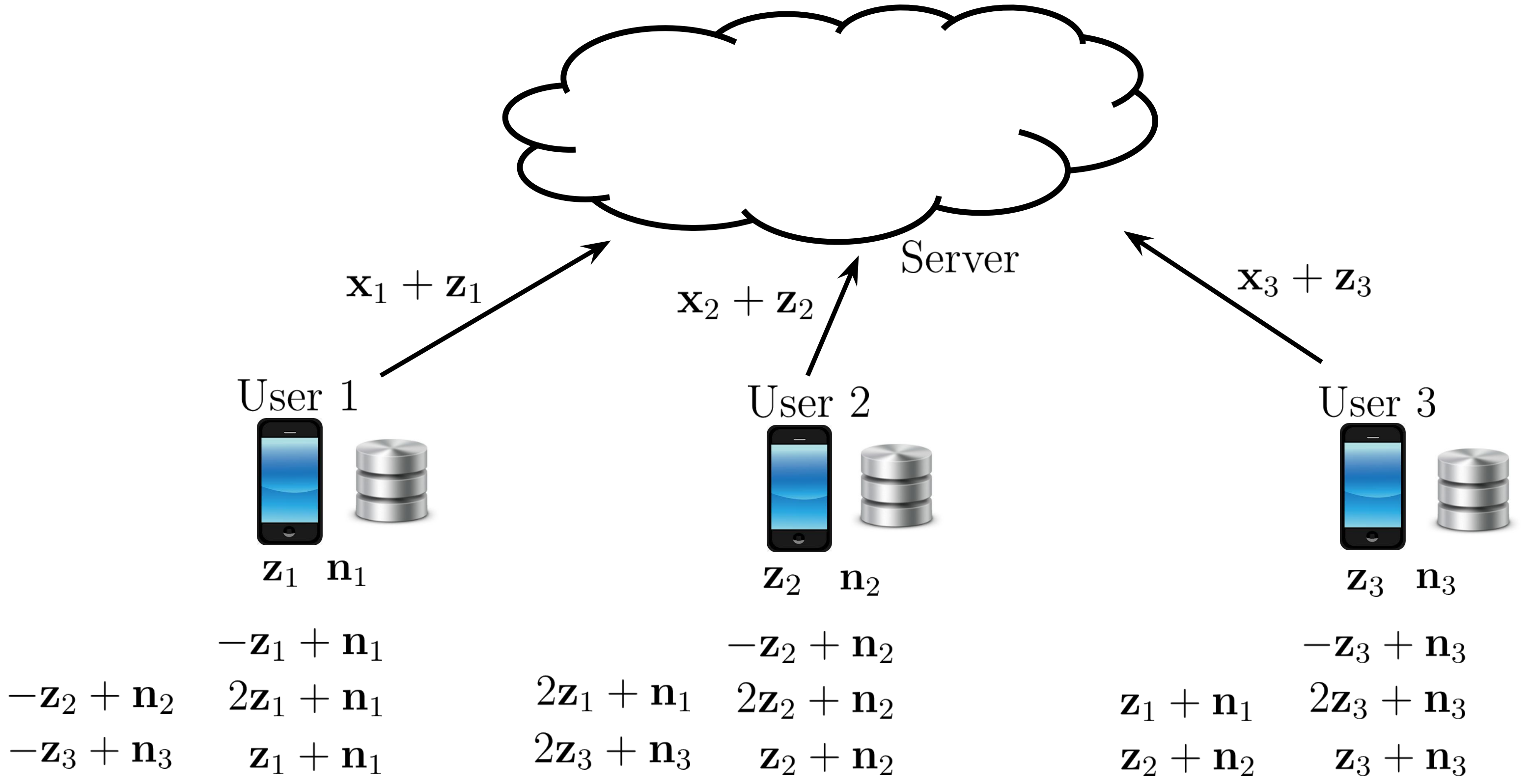
Example (LightSecAgg)

1) Offline encoding and sharing of local masks



Example (LightSecAgg)

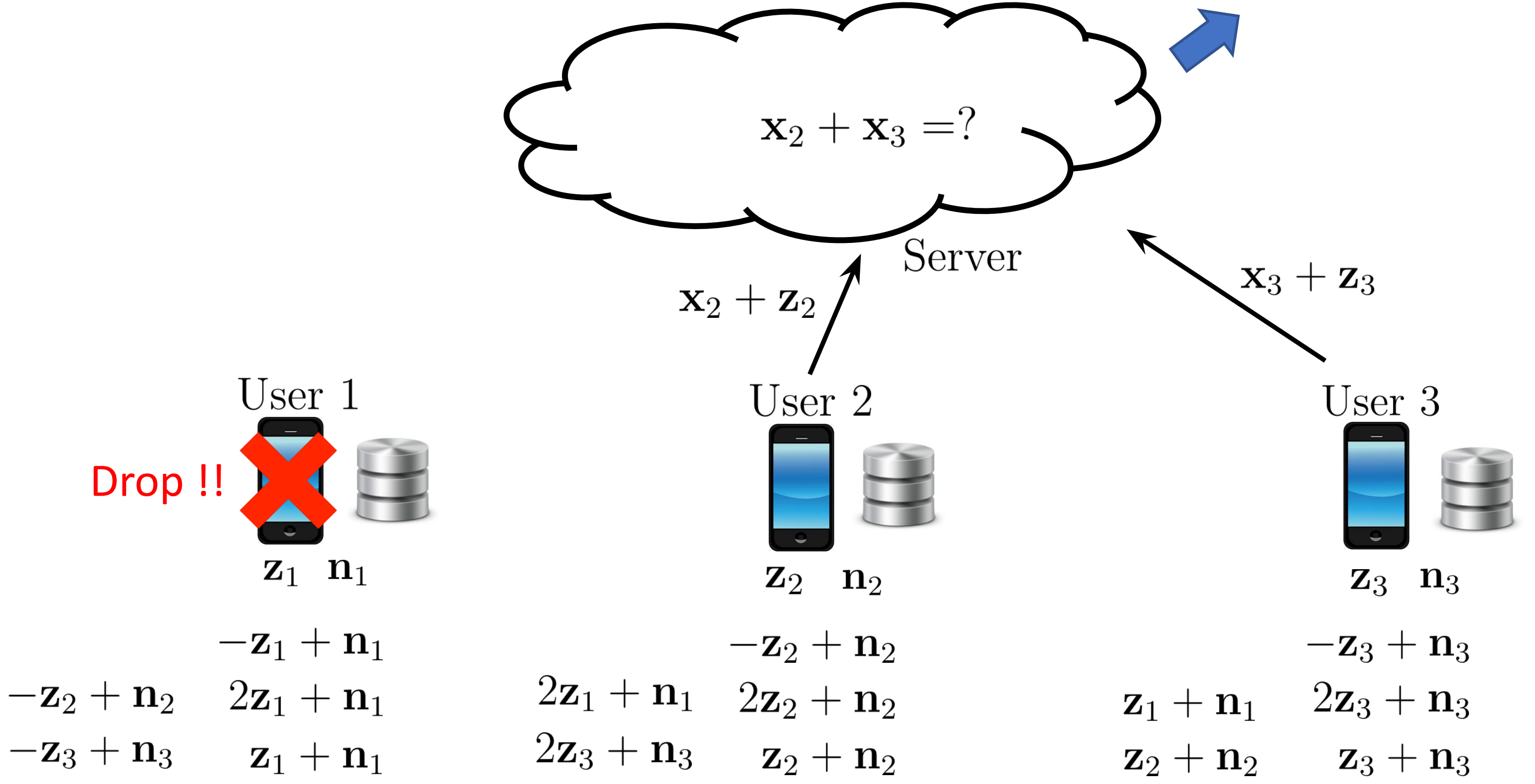
2) Masking and uploading of local models



Example (LightSecAgg)

2) Masking and uploading of local models

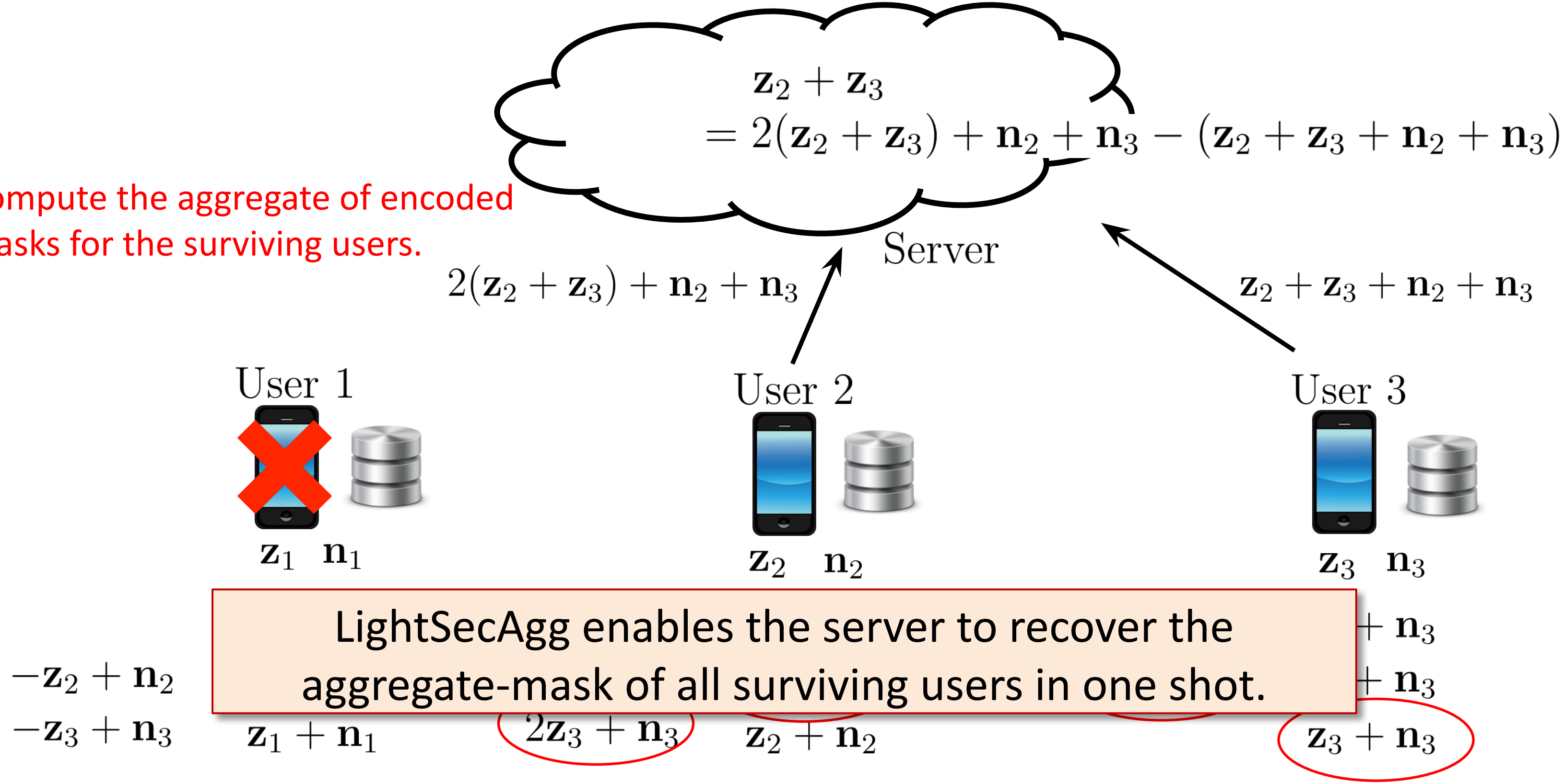
$$\mathbf{x}_2 + \mathbf{x}_3 = (\mathbf{x}_2 + \mathbf{z}_2) + (\mathbf{x}_3 + \mathbf{z}_3) - \color{red}{(\mathbf{z}_2 + \mathbf{z}_3)}$$



Example (LightSecAgg)

3) One-shot aggregate-model recovery

Compute the aggregate of encoded masks for the surviving users.



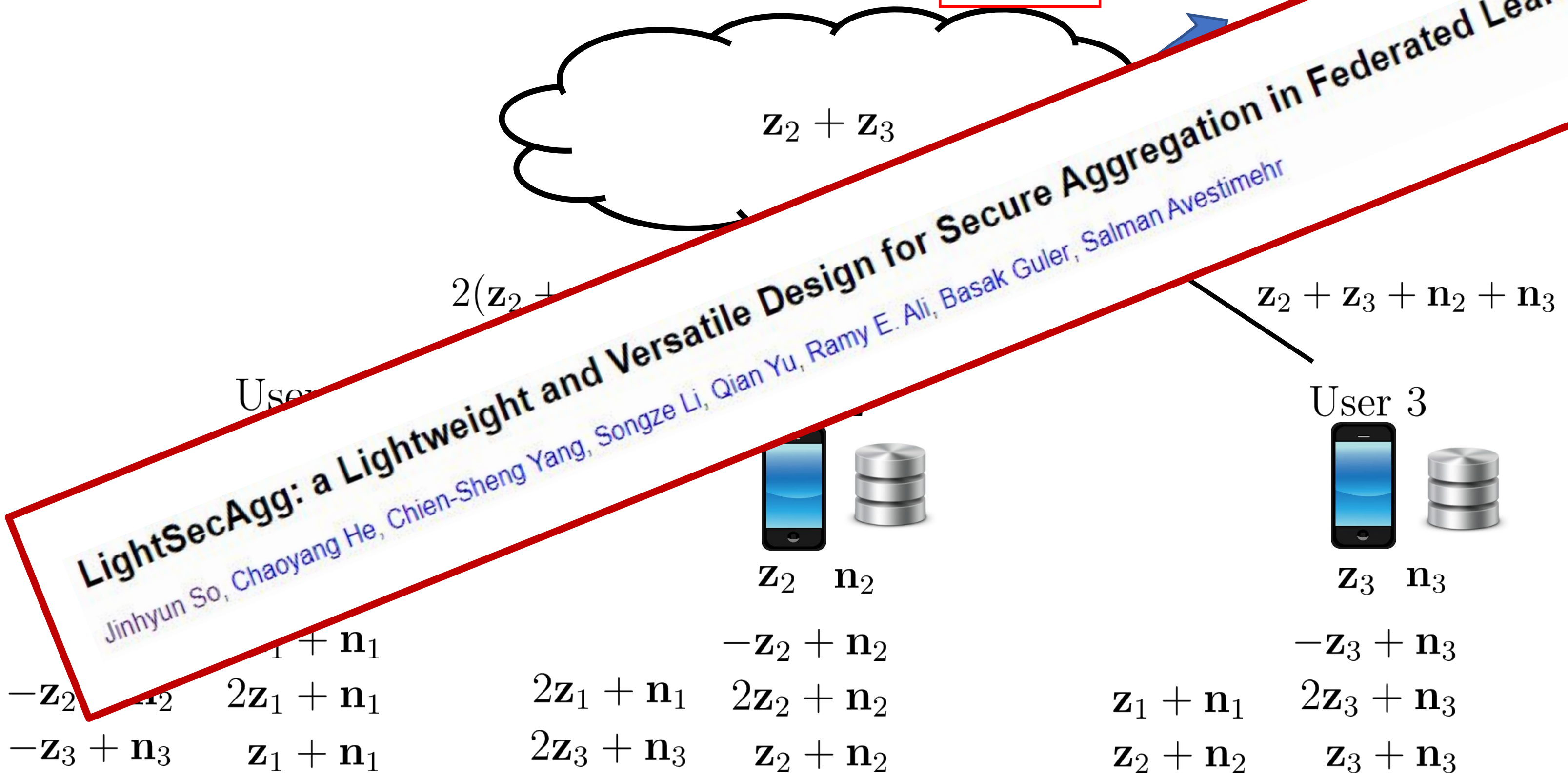
LightSecAgg enables the server to recover the aggregate-mask of all surviving users in one shot.

Example (LightSecAgg)

3) One-shot aggregate-model recovery

Aggregate-model

$\boxed{\mathbf{x}_2 + \mathbf{x}_3} = (\mathbf{x}_2 + \mathbf{z}_2) + \ldots + \mathbf{z}_3)$



LightSecAgg: a Lightweight and Versatile Design for Secure Aggregation in Federated Learning

Jinhyun So, Chaoyang He, Chien-Sheng Yang, Songze Li, Qian Yu, Ramy E. Ali, Basak Guler, Salman Avestimehr

Theoretical Guarantees

- Complexity comparison between SecAgg, SecAgg+ and LightSecAgg:
  - d: model size.
  - s: length of the secret keys.

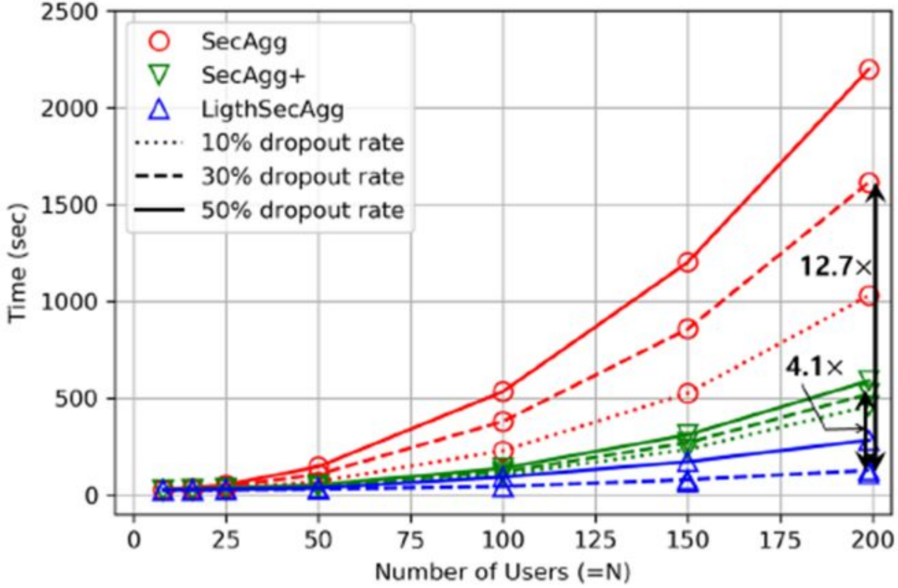
	SecAgg	SecAgg+	LightSecAgg
Offline communication per user	$O(sN)$	$O(s \log N)$	$O(d)$
Offline computation per user	$O(dN + sN^2)$	$O(d \log N + s \log^2 N)$	$O(d \log N)$
Online communication per user	$O(d + sN)$	$O(d + s \log N)$	$O(d)$
Online communication at server	$O(dN + sN^2)$	$O(dN + sN \log N)$	$O(dN)$
Online computation per user	$O(d)$	$O(d)$	$O(d)$
Reconstruction complexity at server	$O(dN^2)$	$O(dN \log N)$	$O(d \log N)$

LightSecAgg significantly improves the computation efficiency at the server during aggregation.

Experiments

- Experiment setup:
  - Amazon EC2 cloud using m3.medium machine instances
  - Four different machine learning tasks
  - Communication using the MPI4Py message passing interface on Python
  - Each user drops with a fixed dropout rate $p = 0.1$, $p = 0.3$, and $p = 0.5$

Experiments



Protocols	Phase	Non-overlapped		
		$p = 10\%$	$p = 30\%$	$p = 50\%$
LightSecAgg	Offline	69.3	69.0	191.2
	Training	22.8	22.8	22.8
	Uploading	12.4	12.2	21.6
	Recovery	40.9	40.7	64.5
	Total	145.4	144.7	300.1
SecAgg	Offline	95.6	98.6	102.6
	Training	22.8	22.8	22.8
	Uploading	10.7	10.9	11.0
	Recovery	911.4	1499.2	2087.0
	Total	1047.5	1631.5	2216.4
SecAgg+	Offline	67.9	68.1	69.2
	Training	22.8	22.8	22.8
	Uploading	10.7	10.8	10.7
	Recovery	379.1	436.7	495.5
	Total	470.5	538.4	608.2

LightSecAgg achieves a performance gain of up to 12.7x !!

Experiments

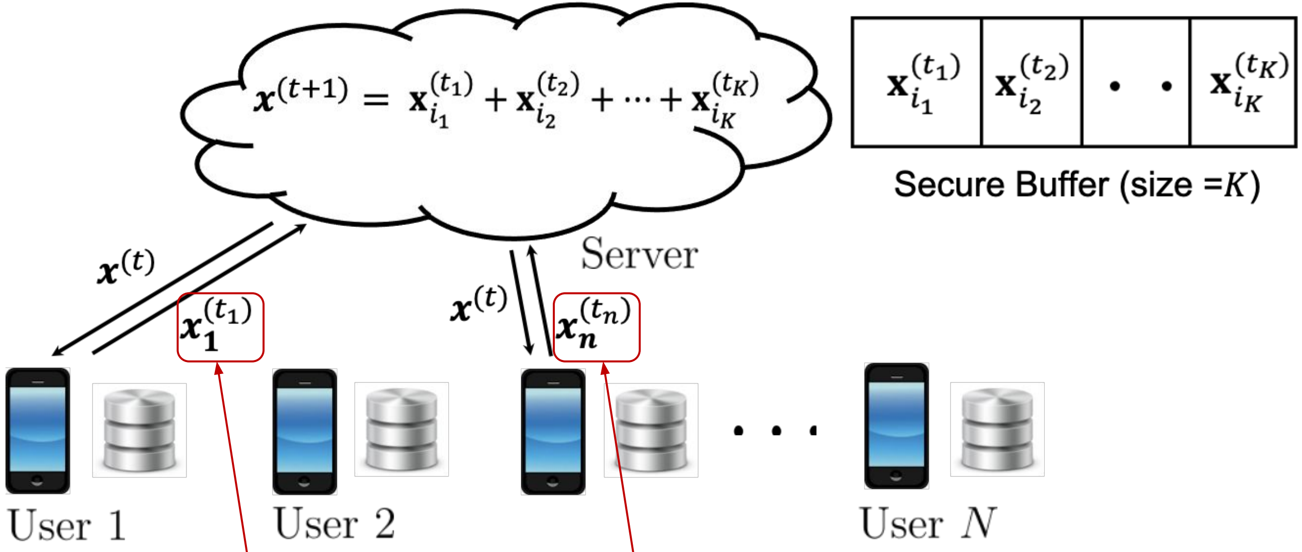
Table 2: Summary of four implemented machine learning tasks and performance gain of `LightSecAgg` with respect to `SecAgg` [4] and `SecAgg+` [2]. All learning tasks are for image classification. MNIST, FEMNIST and CIFAR-100 are low-resolution datasets, while images in GLD-23K are high resolution, which cost much longer training time for one mini-batch; LR and CNN are shallow models, but MobileNetV3 and EfficientNet-B0 are much larger models, but they are tailored for efficient edge training and inference.

No.	Dataset	Model	Model Size (d)	Gain	
				Non-overlapped	Overlapped
1	MNIST [14]	Logistic Regression	7,850	6.7×, 2.5×	8.0×, 2.9×
2	FEMNIST [5]	CNN [17]	1,206,590	11.3×, 3.7×	12.7×, 4.1×
3	CIFAR-100 [13]	MobileNetV3 [11]	3,111,462	7.6×, 2.8×	9.5×, 3.3×
4	GLD-23K [27]	EfficientNet-B0 [24]	5,288,548	3.3×, 1.6×	3.4×, 1.7×

LightSecAgg can survive and speedup the training of large deep neural network models on high resolution image datasets.

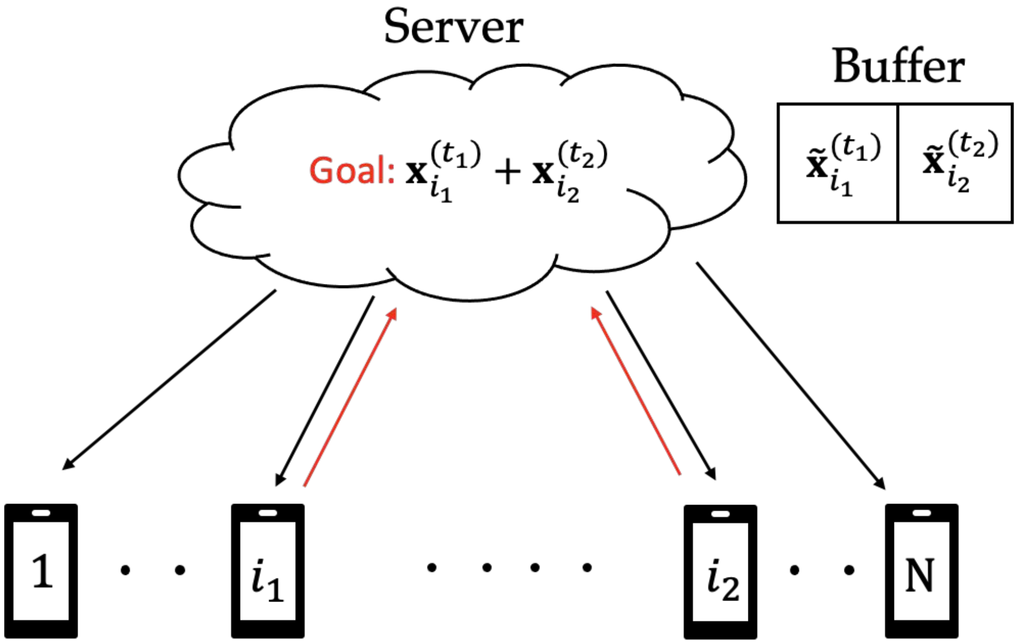
Asynchronous Federated Learning

- There is a growing interest for using **asynchronous FL** to make the system scalable



Challenge: mismatch in staleness!

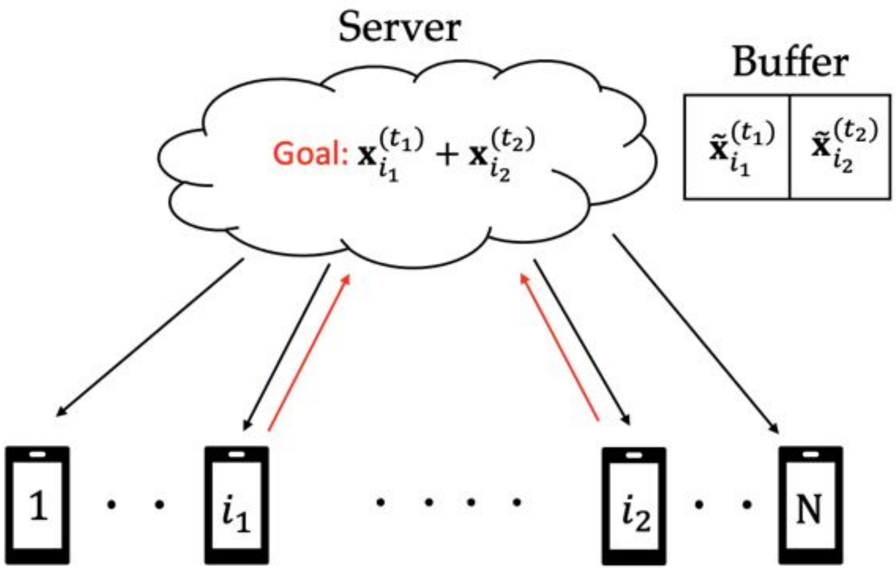
Incompatibility of SecAgg with Asynchronous FL



$$\underbrace{\tilde{\mathbf{x}}_{i_1}^{(t_1)} = \mathbf{x}_{i_1}^{(t_1)} + \mathbf{z}_{i_1,i_2}^{(t_1)} + \cdots \qquad \tilde{\mathbf{x}}_{i_2}^{(t_2)} = \mathbf{x}_{i_2}^{(t_2)} - \mathbf{z}_{i_1,i_2}^{(t_2)}}_{} + \cdots$$

The masks do not cancel out due to the mismatch in staleness!

Asynchronous LightSecAgg



$$\sum_{i\in S^{(t)}} \tilde{\boldsymbol{x}}_i^{(t_i)} = \sum_{i\in S^{(t)}} \boldsymbol{x}_i^{(t_i)} + \boxed{\sum_{i\in S^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

LightSecAgg is compatible as it enables **one-shot recovery of sum of masks** by utilizing MDS structure, even though the masks are generated in different training rounds!

Concluding Remarks

- We propose a new perspective for secure model aggregation in FL, by turning the focus from “pairwise random-seed reconstruction of the dropped users” to “one-shot aggregate-mask reconstruction of the surviving users”.

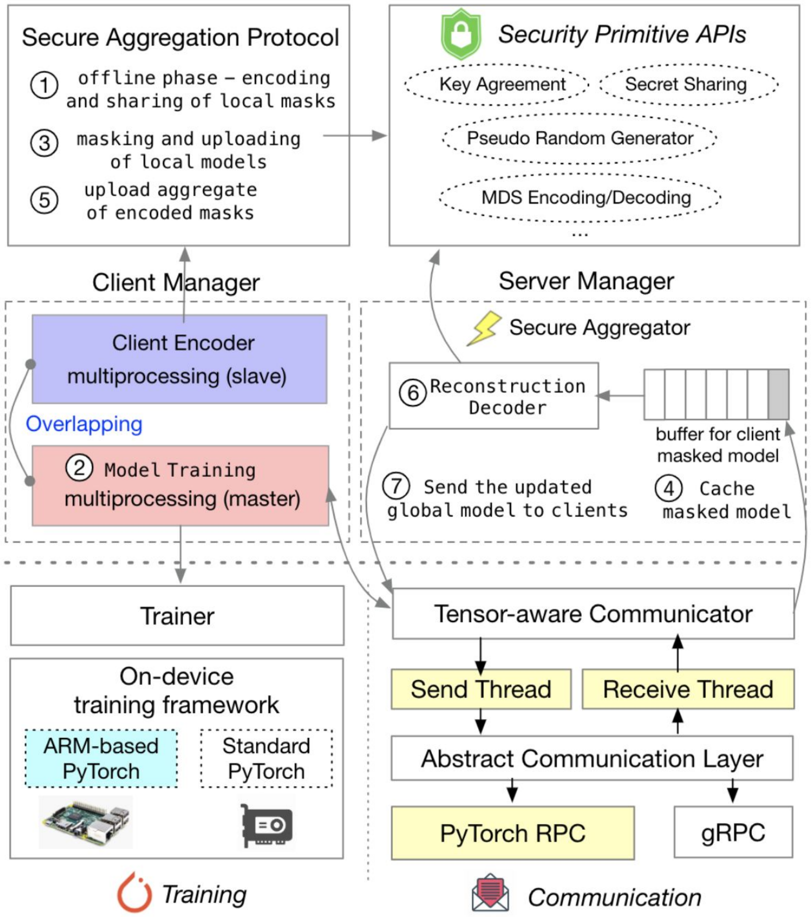
- We propose LightSecAgg that provides the same level of privacy and dropout-resiliency guarantees as the state-of-the-art while substantially reducing the aggregation complexity.

- LightSecAgg is the first secure aggregation protocol that can be applied to asynchronous FL.

Appendix

Appendix 1. System-level Optimization

Overview of the System Design



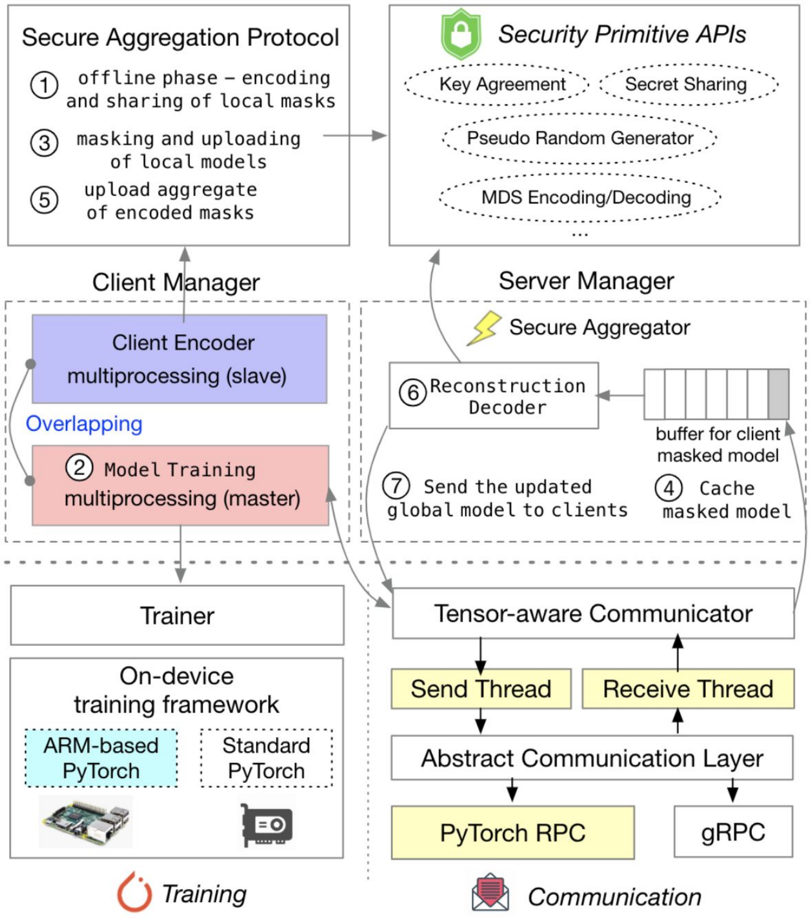
Design Goals:

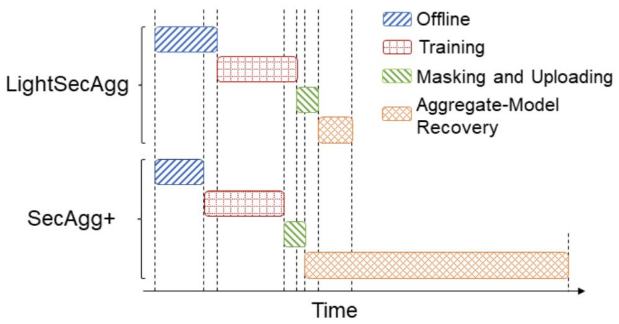
1. Make the system API friendly to pure ML researchers who may not have expertise in SA/Security.

2. Reduce the cost of encoding and decoding at the edge

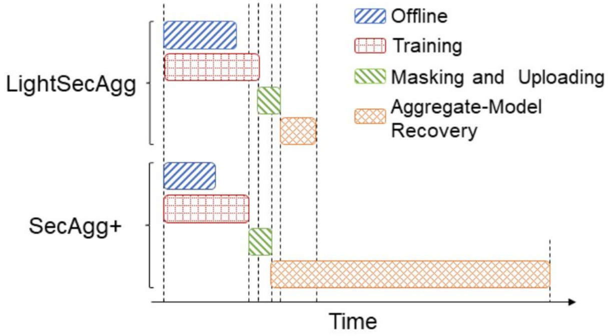
3. Optimize the communication backend, making it Torch Tensor-aware

1. Parallelization of offline phase and model training



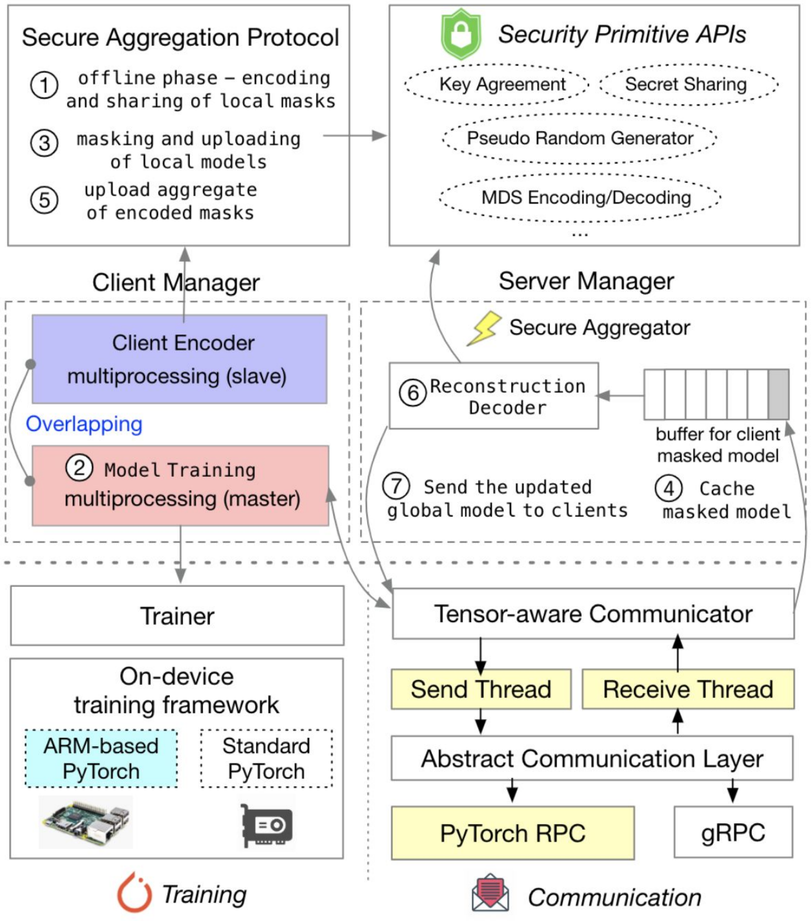


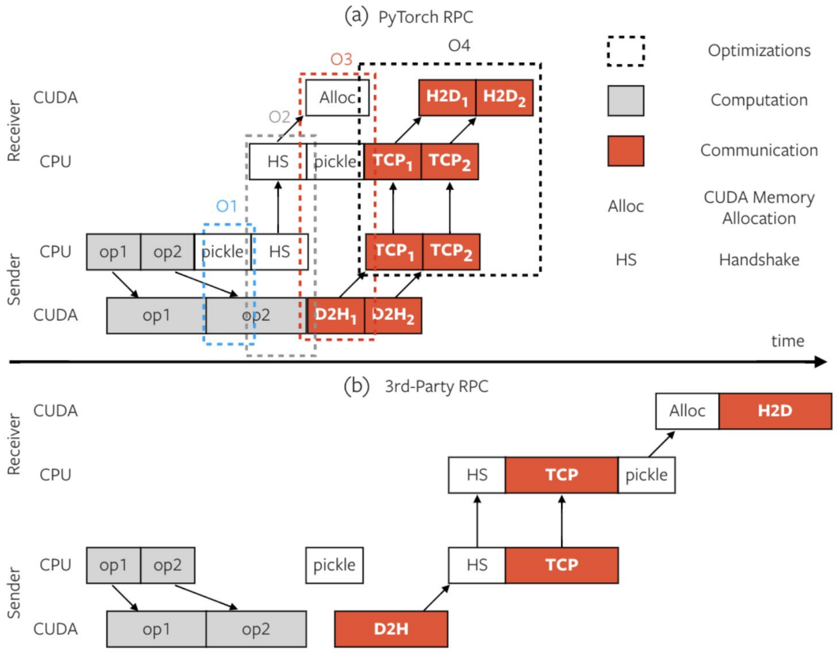
(a) Non-overlapped



(b) Overlapped

2. Tensor-aware RPC (Remote Procedure Call)



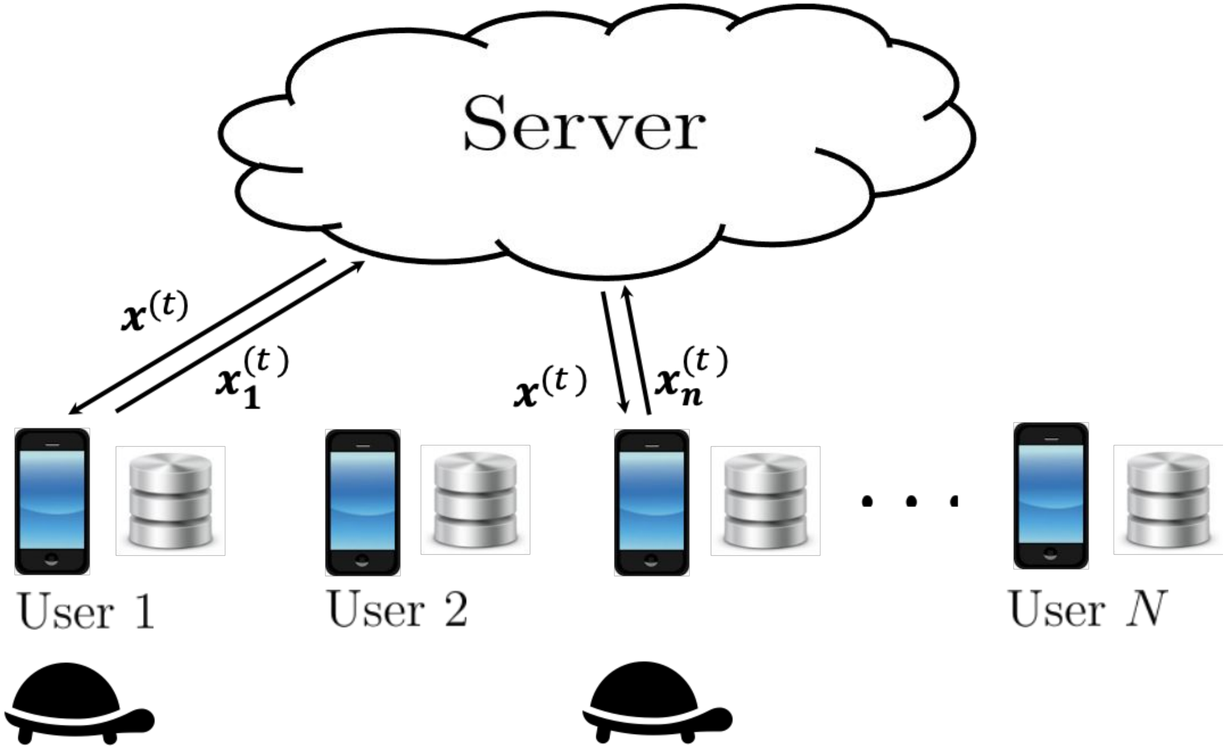


Comparison with Third-Party RPC

Appendix 2.
LightSecAgg for Asynchronous Federated Learning

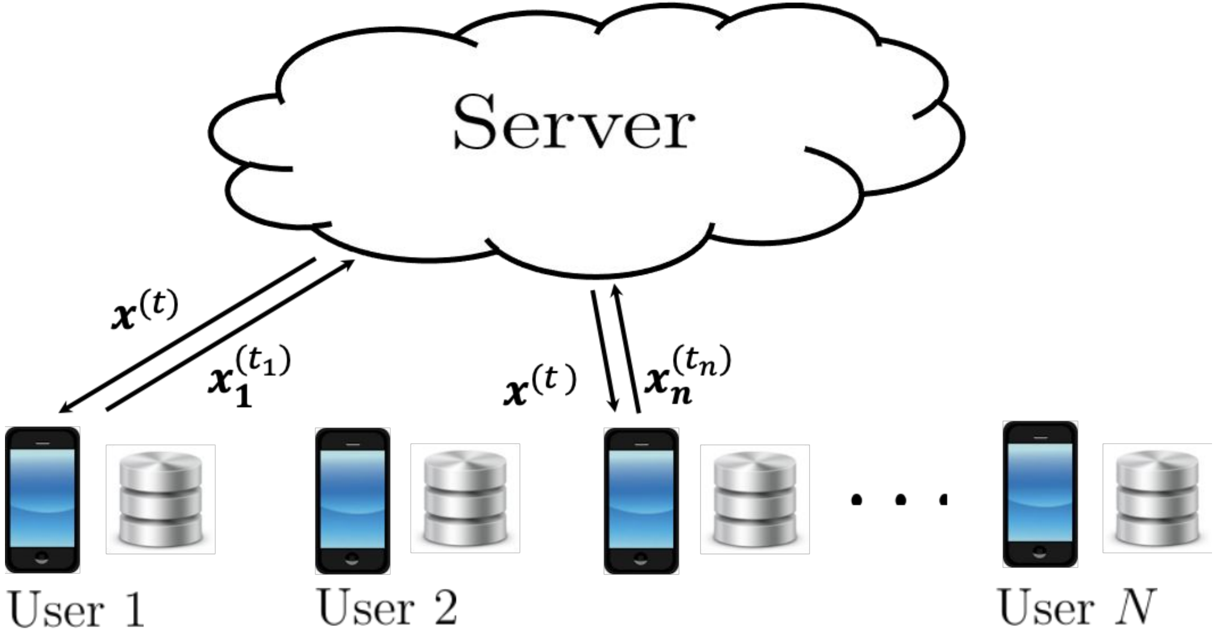
Asynchronous FL

- Synchronous FL suffers from stragglers!



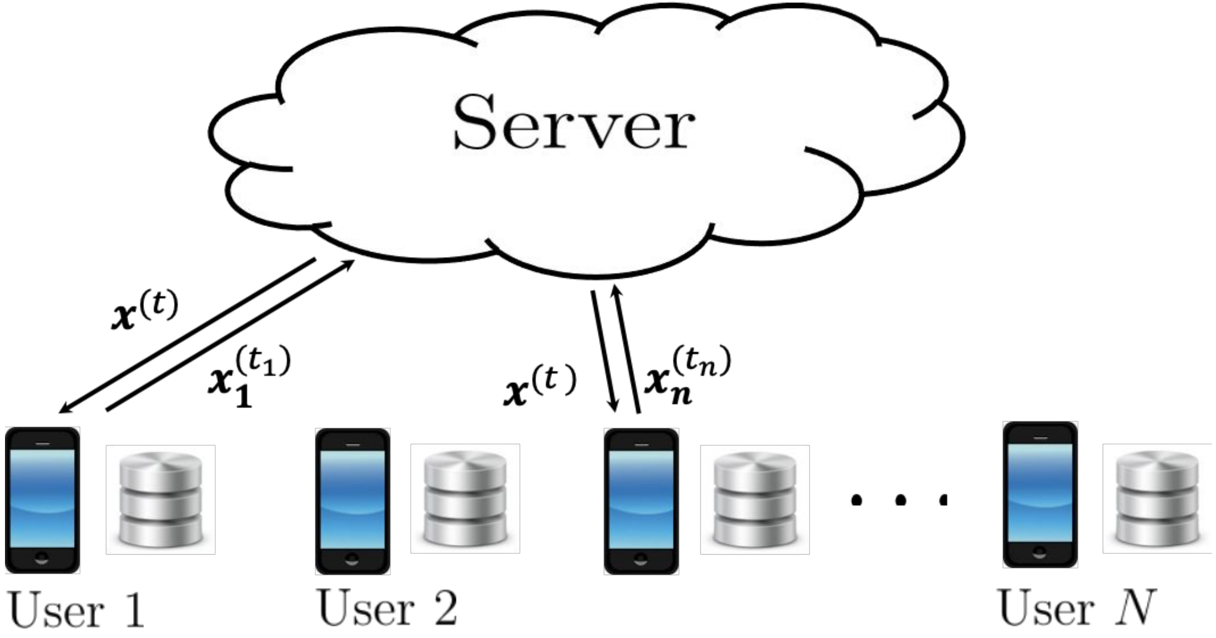
Asynchronous FL

- Updates are not synchronized.
- Each local model received updates the global model.



Asynchronous FL

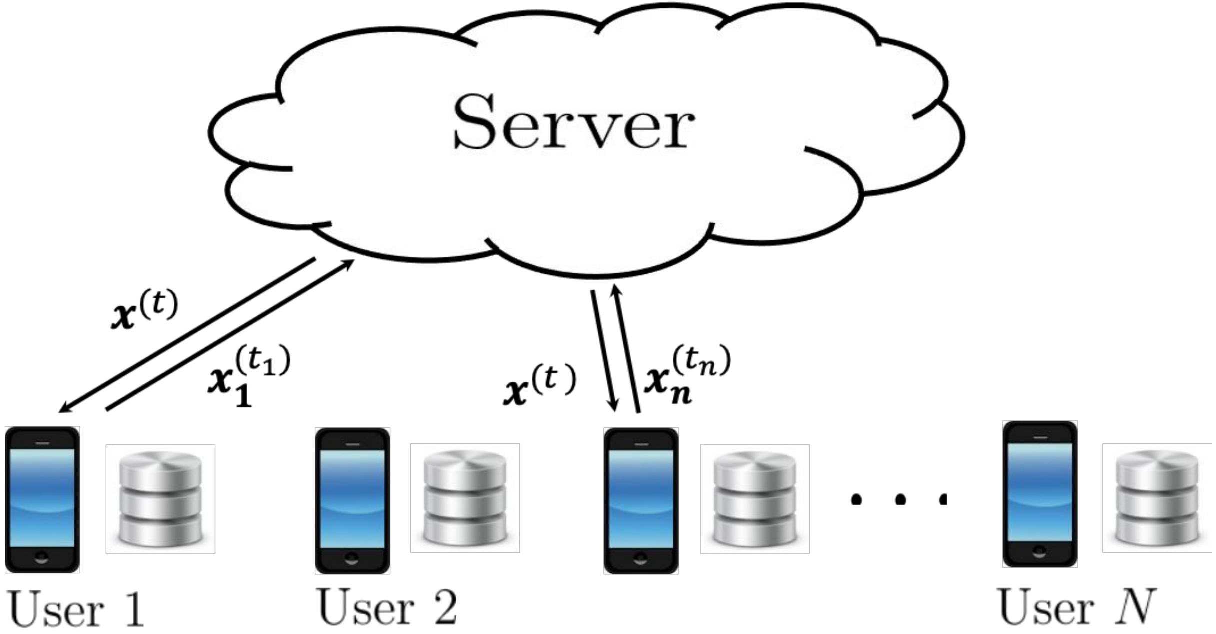
- Updates are not synchronized.
- Each local model received updates the global model.



Not compatible with secure aggregation!

Asynchronous FL

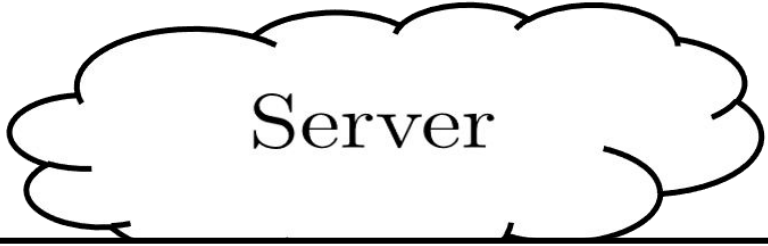
- Updates are not synchronized.
- Each local model received updates the global model.

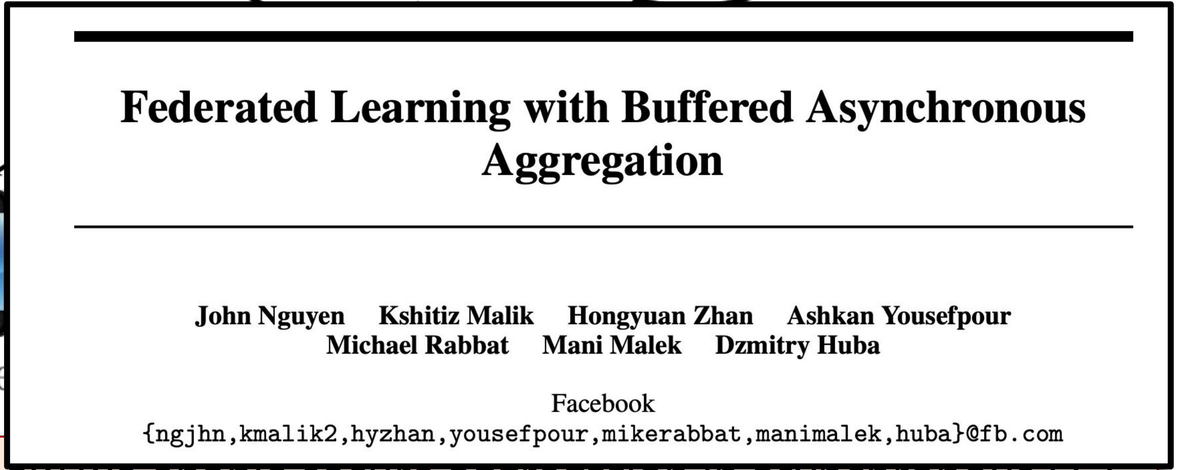


How we enable secure aggregation in asynchronous FL?

Asynchronous FL

- Updates are not synchronized.
- Each local model received updates the global model.





Federated Learning with Buffered Asynchronous Aggregation

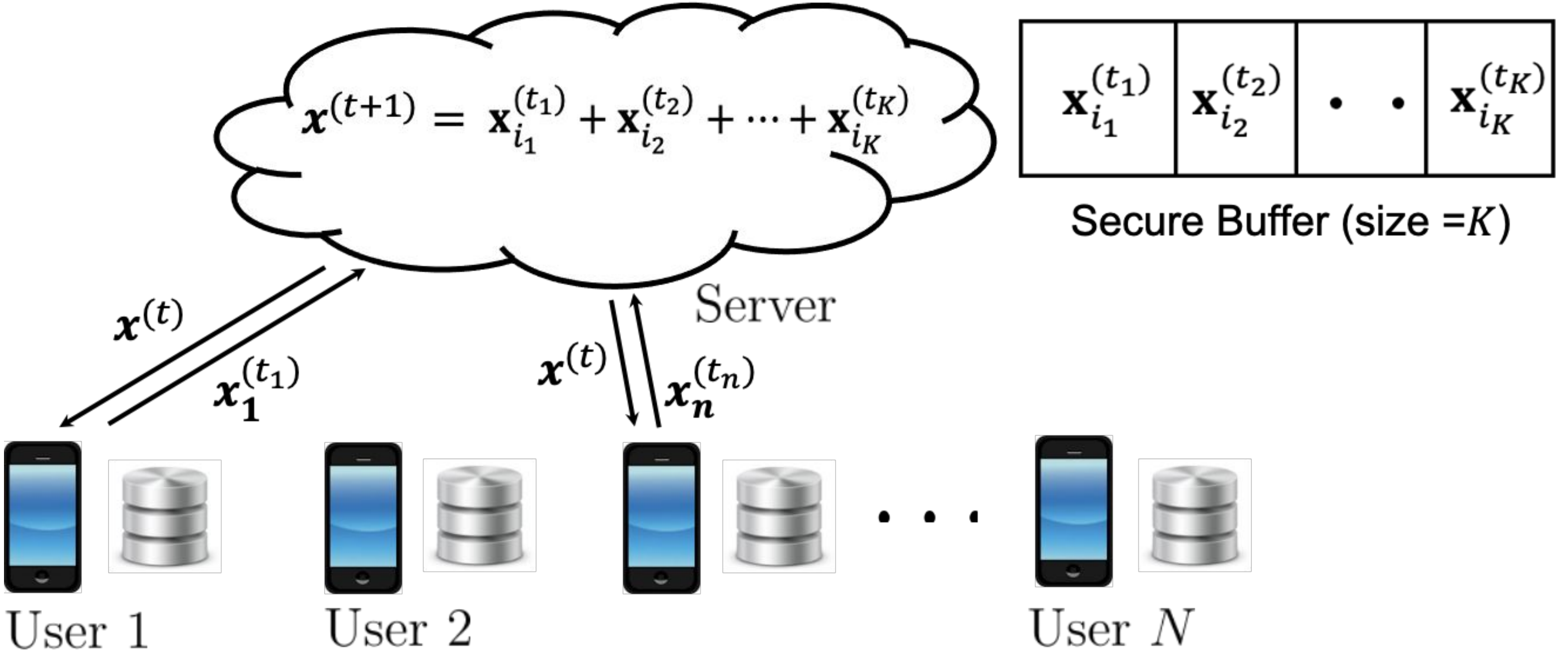
John Nguyen Kshitiz Malik Hongyuan Zhan Ashkan Yousefpour
Michael Rabbat Mani Malek Dzmitry Huba

Facebook
{ngjhn,kmalik2,hyzhan,yousefpour,mikerabbat,manimalek,huba}@fb.com

How we enable secure aggregation in asynchronous FL?

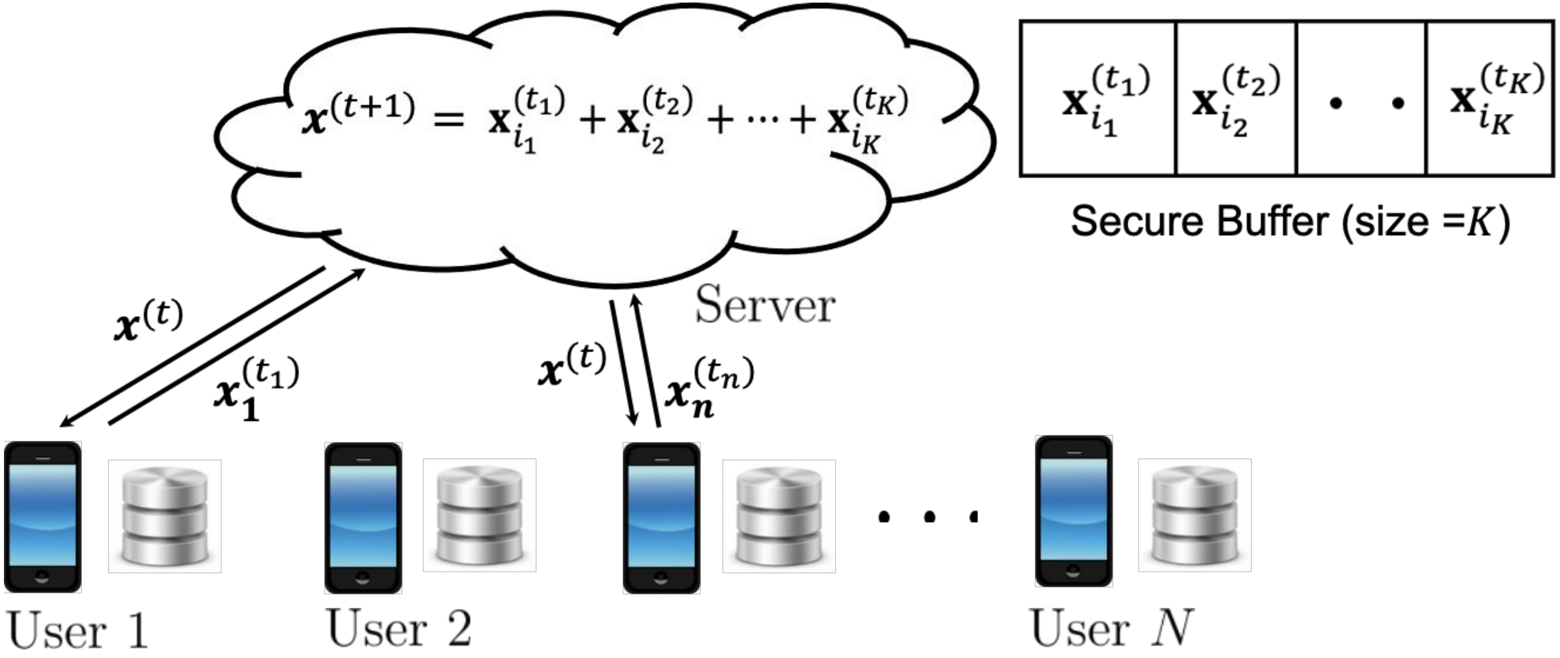
Asynchronous FL

- Typical Asynchronous FL: K=1 (not compatible with secure aggregation)
- Buffered Asynchronous FL (FedBuff): K>1



Asynchronous FL

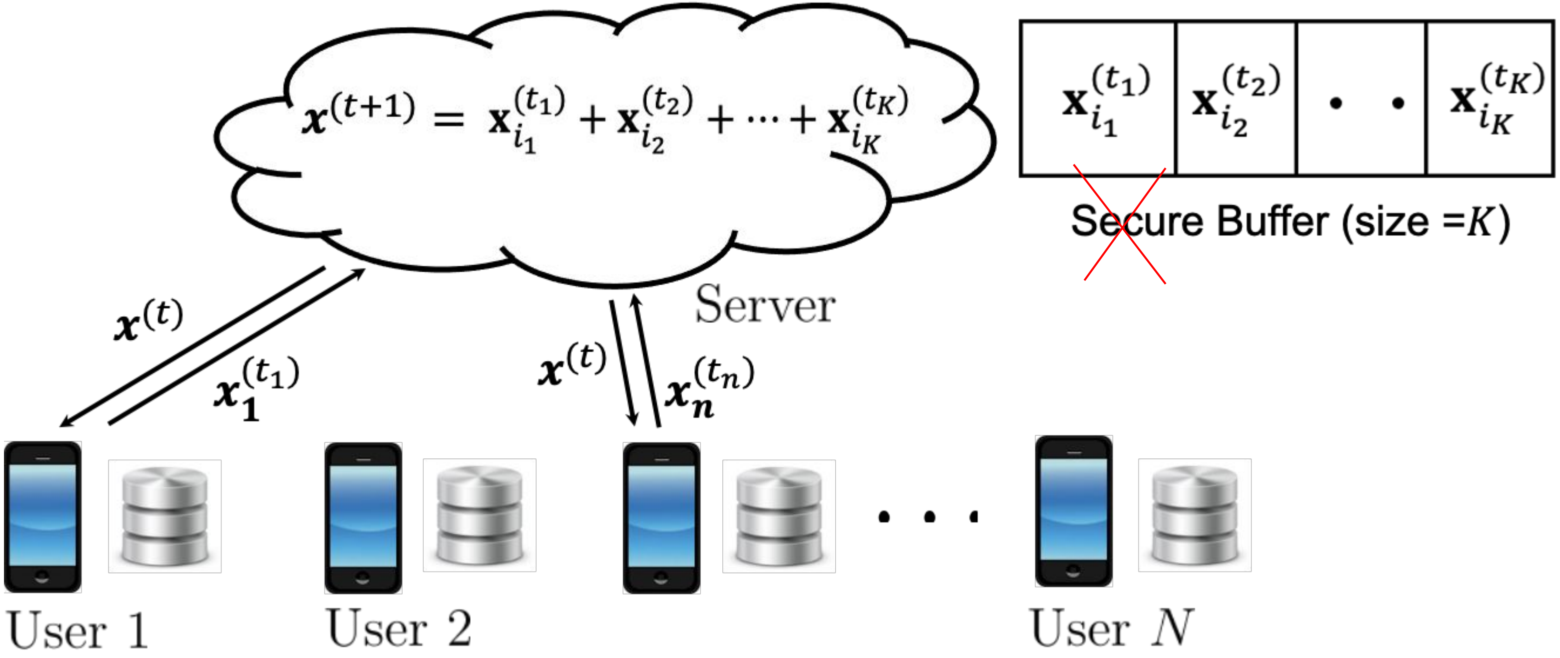
- Typical Asynchronous FL: K=1 (not compatible with secure aggregation)
- Buffered Asynchronous FL (FedBuff): K>1



The TEE-enabled secure buffer, however, has limited memory.
(K must be small!)

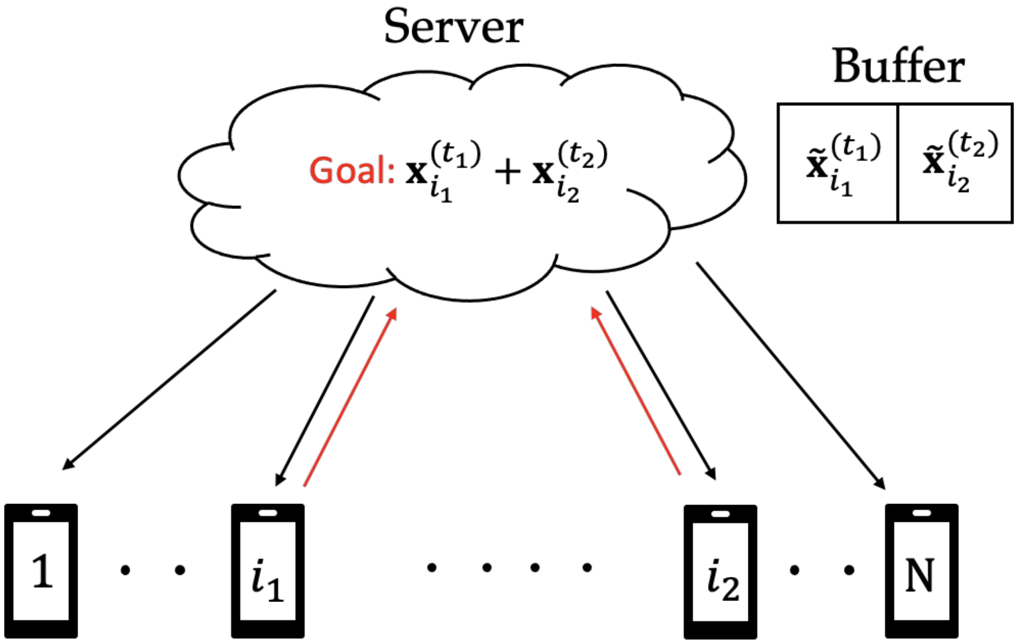
Asynchronous FL

- Typical Asynchronous FL: K=1 (not compatible with secure aggregation)
- Buffered Asynchronous FL (FedBuff): K>1



LightSecAgg does not require a secure buffer!

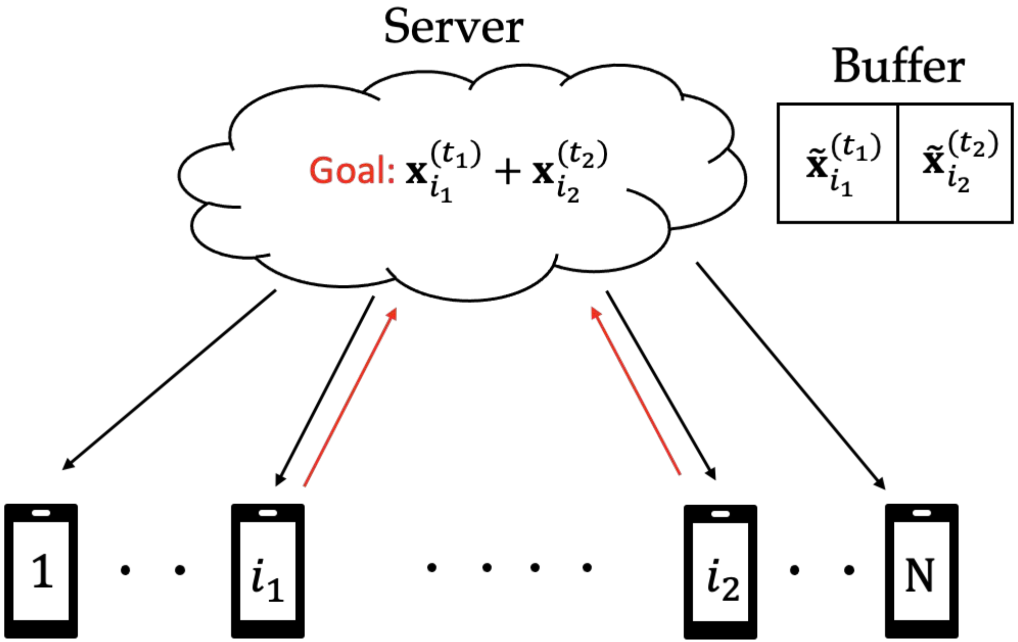
Incompatibility of SecAgg with Asynchronous FL



$$\tilde{\mathbf{x}}_{i_1}^{(t_1)} = \mathbf{x}_{i_1}^{(t_1)} + \mathbf{z}_{i_1,i_2}^{(t_1)} + \cdots \qquad \tilde{\mathbf{x}}_{i_2}^{(t_2)} = \mathbf{x}_{i_2}^{(t_2)} - \mathbf{z}_{i_1,i_2}^{(t_2)} + \cdots$$

The masks do not cancel out due to the mismatch in staleness!

Incompatibility of SecAgg with Asynchronous FL



$$\tilde{\mathbf{x}}_{i_1}^{(t_1)} = \mathbf{x}_{i_1}^{(t_1)} + \mathbf{z}_{i_1,i_2}^{(t_1)} + \cdots \qquad \tilde{\mathbf{x}}_{i_2}^{(t_2)} = \mathbf{x}_{i_2}^{(t_2)} - \mathbf{z}_{i_1,i_2}^{(t_2)} + \cdots$$

How to design the masks to cancel out even if they belong to different rounds?

Asynchronous LightSecAgg

Key objective: Design the masks such that they cancel out even if they belong to different training rounds.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta_g \boldsymbol{g}^{(t)} \text{ where } \boldsymbol{g}^{(t)} = \sum_{i \in S^{(t)}} \widetilde{\boldsymbol{x}}_i^{(t;t_i)} = \sum_{i \in S^{(t)}} \overline{\boldsymbol{x}}_i^{(t;t_i)} + \boxed{\sum_{i \in S^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

LightSecAgg is compatible as it does not use pair-wise masking!

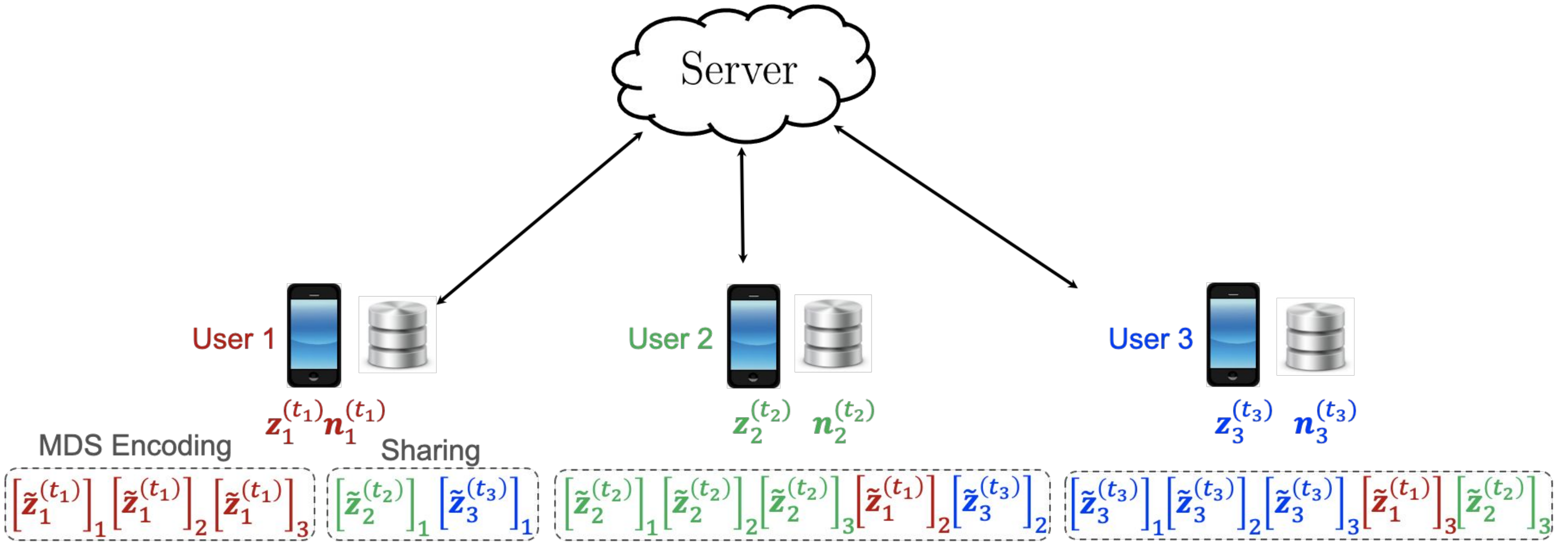
Asynchronous LightSecAgg

Key objective: Design the masks such that they cancel out even if they belong to different training rounds.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta_g \boldsymbol{g}^{(t)} \text{ where } \boldsymbol{g}^{(t)} = \sum_{i \in S^{(t)}} \tilde{\boldsymbol{x}}_i^{(t;t_i)} = \sum_{i \in S^{(t)}} \bar{\boldsymbol{x}}_i^{(t;t_i)} + \boxed{\sum_{i \in S^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

Step 1. Offline encoding and sharing of local masks



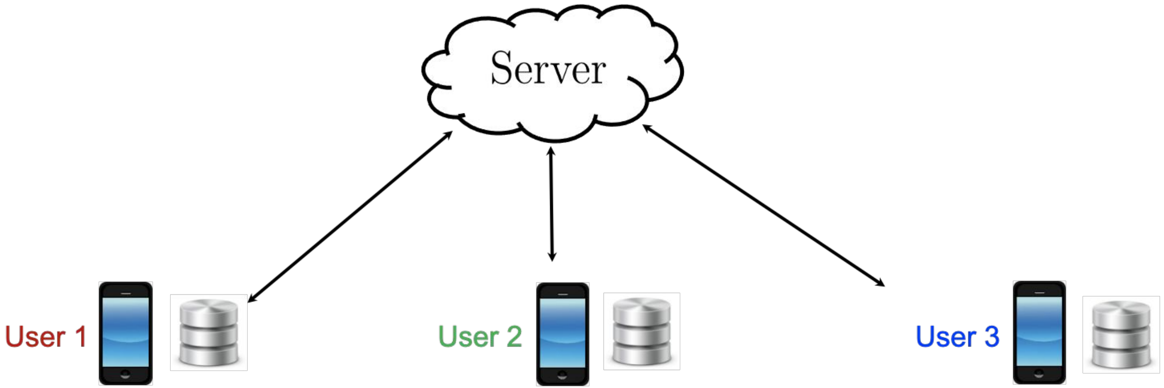
Asynchronous LightSecAgg

Key objective: Design the masks such that they cancel out even if they belong to different training rounds.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta_g \boldsymbol{g}^{(t)} \text{ where } \boldsymbol{g}^{(t)} = \sum_{i \in S^{(t)}} \widetilde{\boldsymbol{x}}_i^{(t;t_i)} = \sum_{i \in S^{(t)}} \overline{\boldsymbol{x}}_i^{(t;t_i)} + \boxed{\sum_{i \in S^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

Step 2. Quantization & Masking



$\widetilde{\boldsymbol{x}}_1^{(t;t_1)} = \overline{\boldsymbol{x}}_1^{(t;t_1)} + \boldsymbol{z}_1^{(t_1)}$ $\widetilde{\boldsymbol{x}}_2^{(t;t_2)} = \overline{\boldsymbol{x}}_2^{(t;t_2)} + \boldsymbol{z}_2^{(t_2)}$ $\widetilde{\boldsymbol{x}}_3^{(t;t_3)} = \overline{\boldsymbol{x}}_3^{(t;t_3)} + \boldsymbol{z}_3^{(t_3)}$

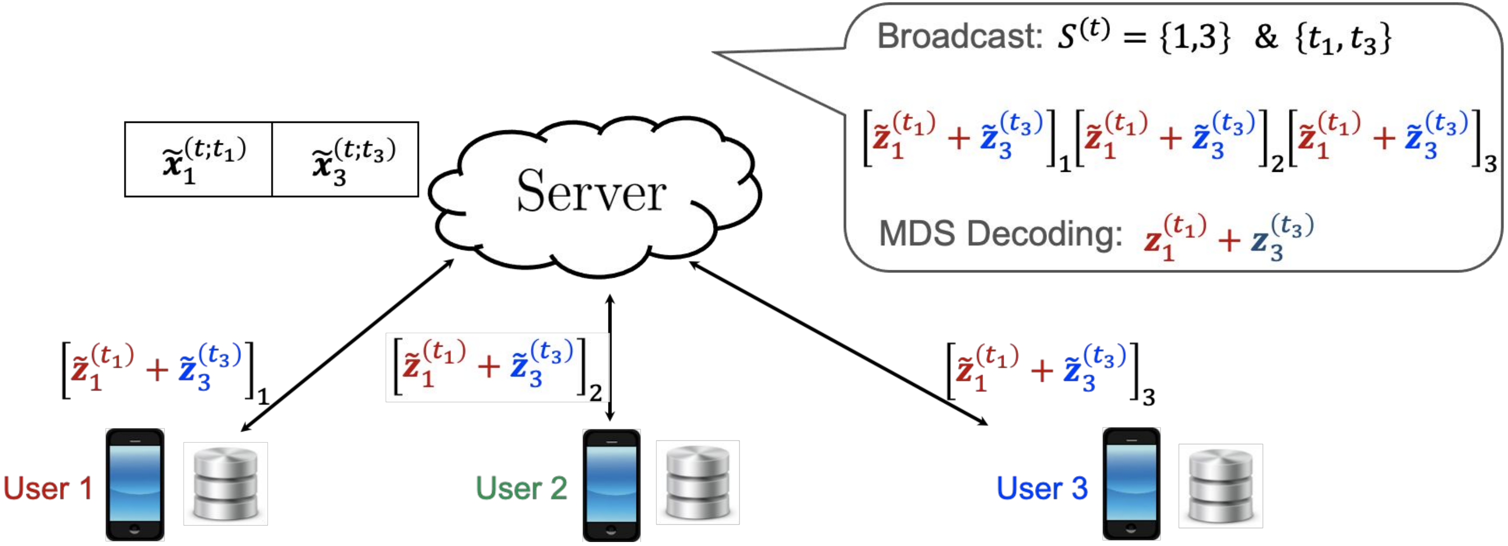
Asynchronous LightSecAgg

Key objective: Design the masks such that they cancel out even if they belong to different training rounds.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta_g \boldsymbol{g}^{(t)} \text{ where } \boldsymbol{g}^{(t)} = \sum_{i \in \mathcal{S}^{(t)}} \tilde{\boldsymbol{x}}_i^{(t;t_i)} = \sum_{i \in \mathcal{S}^{(t)}} \bar{\boldsymbol{x}}_i^{(t;t_i)} + \boxed{\sum_{i \in \mathcal{S}^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

Step 3. One-Shot Recovery of Aggregate Masks



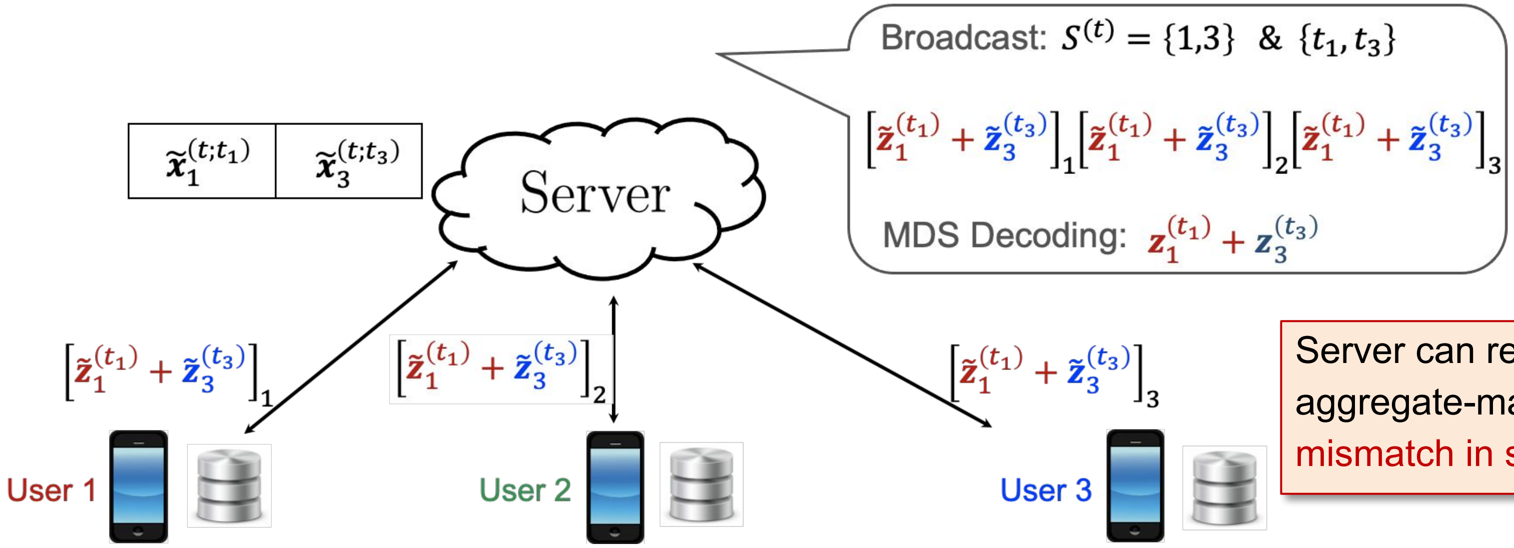
Asynchronous LightSecAgg

Key objective: Design the masks such that they cancel out even if they belong to different training rounds.

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \eta_g \boldsymbol{g}^{(t)} \text{ where } \boldsymbol{g}^{(t)} = \sum_{i \in S^{(t)}} \tilde{\boldsymbol{x}}_i^{(t;t_i)} = \sum_{i \in S^{(t)}} \bar{\boldsymbol{x}}_i^{(t;t_i)} + \boxed{\sum_{i \in S^{(t)}} \boldsymbol{z}_i^{(t_i)}}$$

Our focus

Step 3. One-Shot Recovery of Aggregate Masks



Server can reconstruct the aggregate-masks even if there is mismatch in staleness !