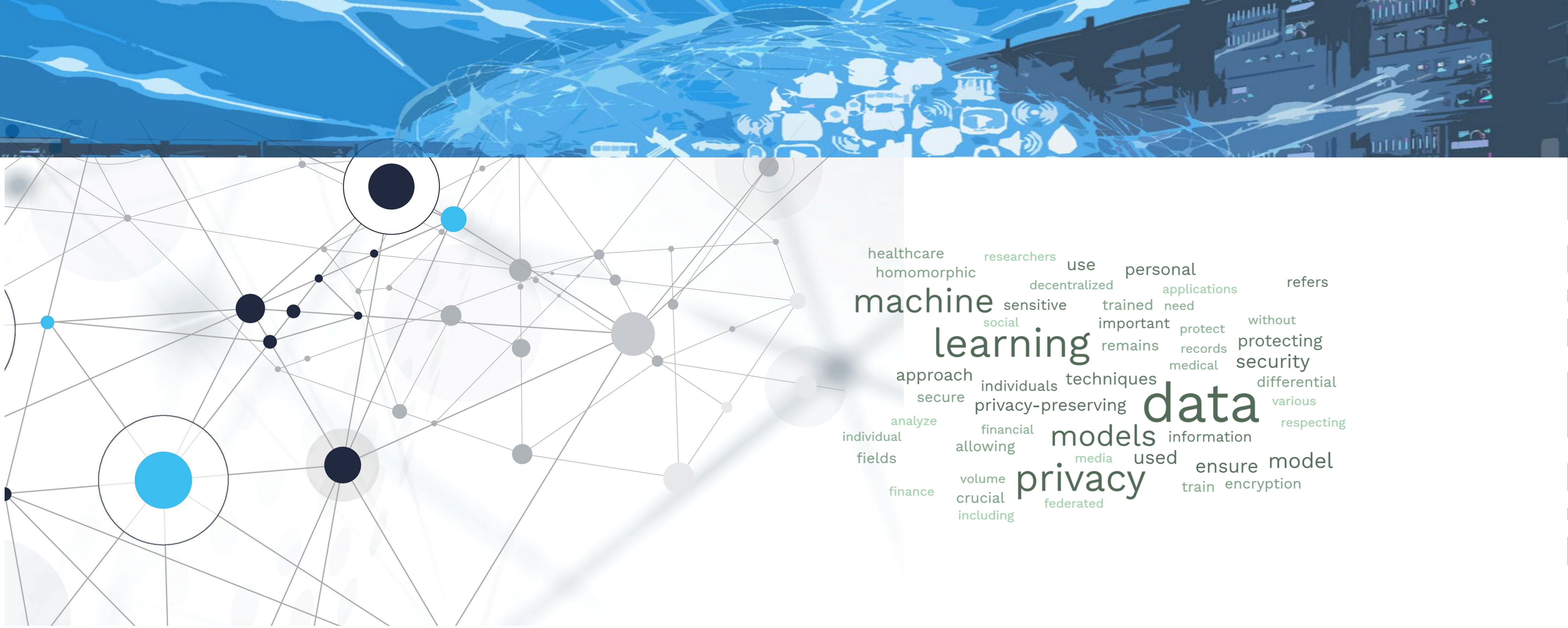


PRIVACY-PRESERVING FEDERATED LEARNING

Dumindu Samaraweera, PhD
Ren-Yi Huang*

*University of South Florida

AGENDA

- What is Federated Learning
- Privacy and security implications of Federated Learning
- Threats, Defenses, and Privacy-preserving Techniques
  - Model Inversion Attack
  - Model Poisoning Attack
- Solution using DP and MPC for Federated Learning

Q&A

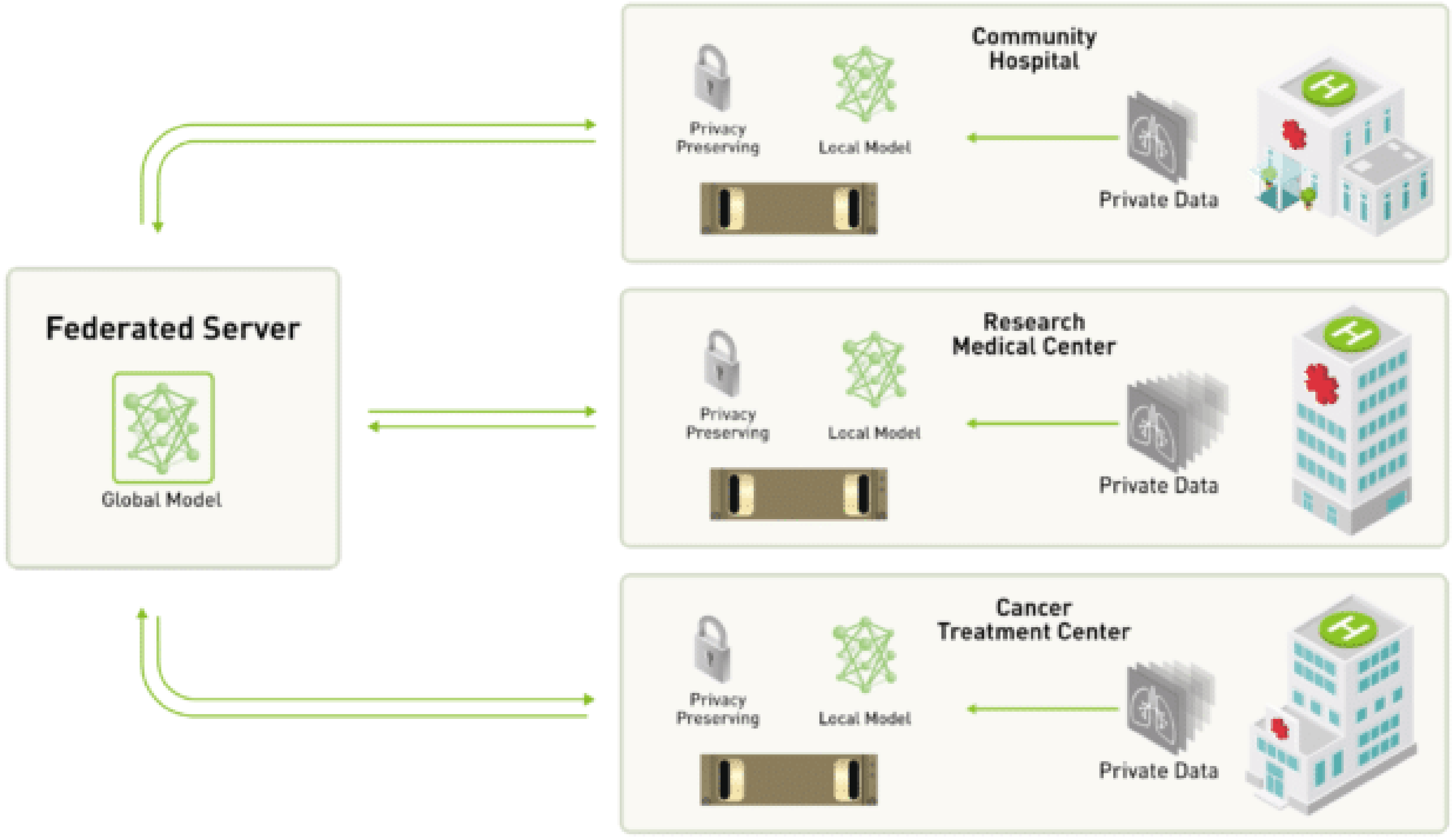
WHAT IS FEDERATED LEARNING

In traditional centralized machine learning, data is collected and sent to a central server for model training, which can raise privacy concerns, especially when dealing with sensitive or personal information.

Federated learning is a machine learning approach that allows multiple parties or devices to collaboratively train a shared machine learning model while keeping their data decentralized and private.

- In federated learning, clients participate in the training process using their local data, without sharing it with other clients or the server.
- Therefore, each client can both contribute to and benefit from a global model.

WHAT IS FEDERATED LEARNING



Melanoma is a type of skin cancer that is less common than some other types of skin cancer, but it is more likely to grow and spread.

However, building an ML model is hard because it suffers from the extreme class-imbalance problem.

Image: https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning/

MAJOR CHALLENGES OF FEDERATED LEARNING

Utility issues

- Non-IID dataset
- Local class imbalance

Privacy issues

- Membership inference
- Training data reconstruction

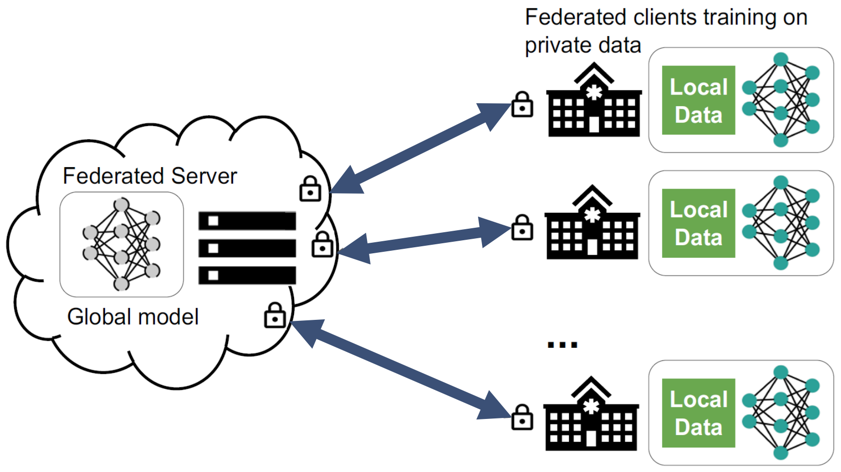
Fairness issues

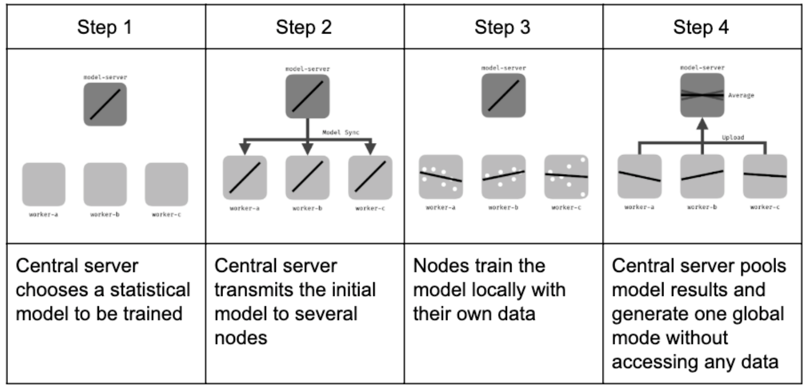
Communication issues

- Communication bandwidth
- Network failure

Level of trust

- Malicious entities in the system





Step 1	Step 2	Step 3	Step 4
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

Image: General step to obtain trained models https://en.wikipedia.org/wiki/Federated_learning

MAJOR CHALLENGES OF FEDERATED LEARNING

Utility issues

- Non-IID dataset
- Local class imbalance

Privacy issues

- Membership inference
- Training data reconstruction

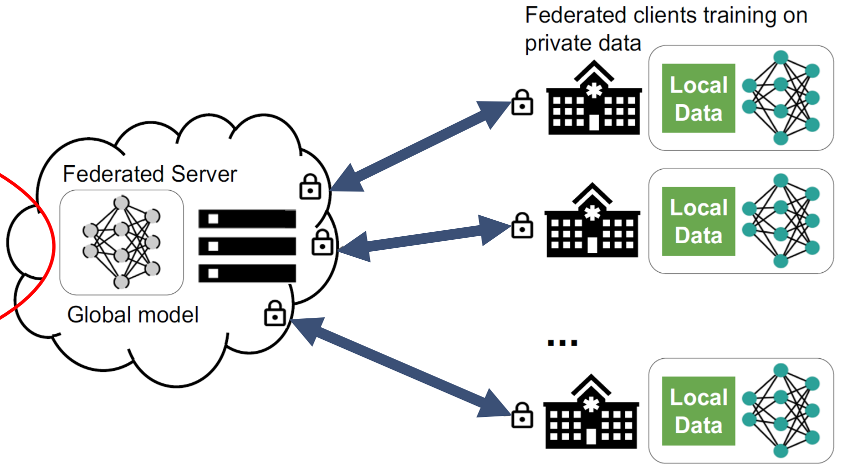
Fairness issues

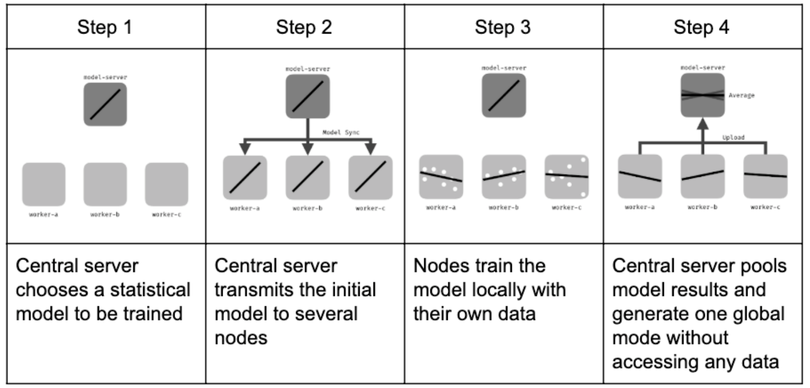
Communication issues

- Communication bandwidth
- Network failure

Level of trust

- Malicious entities in the system





Step 1	Step 2	Step 3	Step 4
Central server chooses a statistical model to be trained	Central server transmits the initial model to several nodes	Nodes train the model locally with their own data	Central server pools model results and generate one global mode without accessing any data

Image: General step to obtain trained models https://en.wikipedia.org/wiki/Federated_learning

PRIVACY AND SECURITY IMPLICATIONS OF FEDERATED LEARNING

Data privacy:

- Even though federated learning is designed to preserve privacy, it is still possible for some sensitive data to be leaked during the training process.
- This can happen if the model is not properly encrypted or if the federated learning system is compromised.

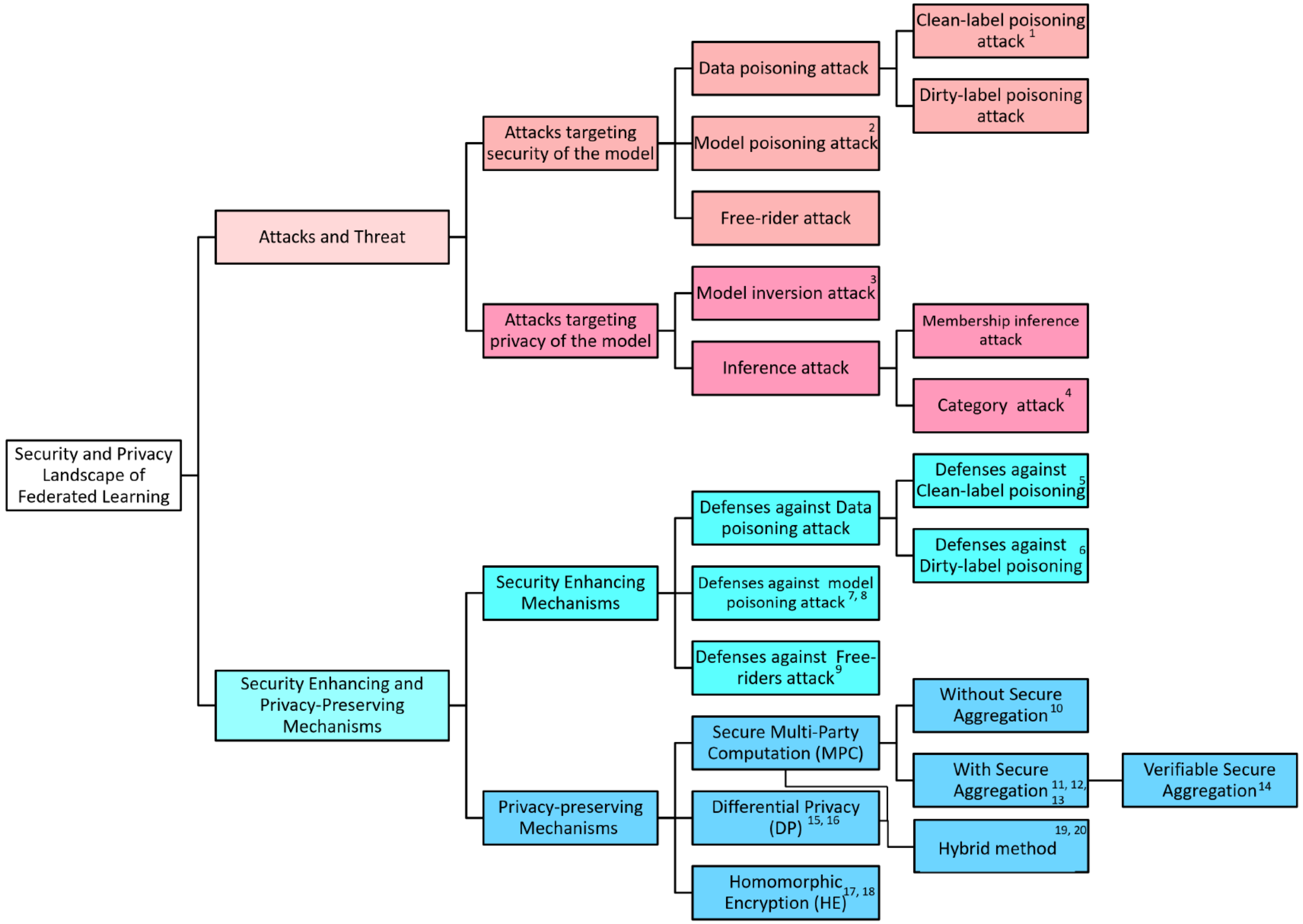
Model poisoning:

- Malicious participants in a federated learning system can try to inject incorrect or malicious data into the system to corrupt the model or steer it in a different direction.

Security vulnerabilities:

- Federated learning systems can be vulnerable to various types of attacks, including data poisoning, man-in-the-middle attacks, and sybil attacks.

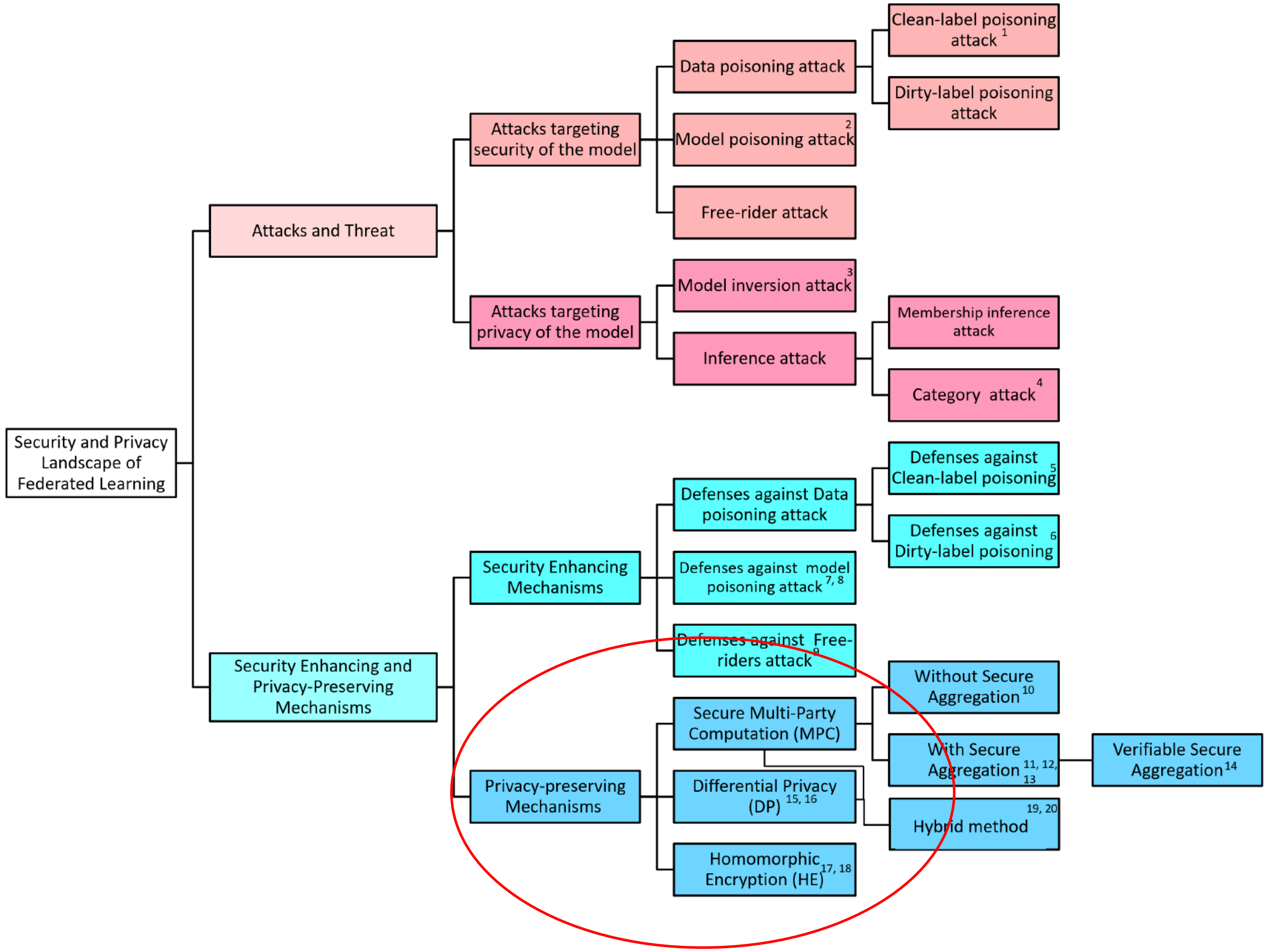
THREATS, DEFENSES, AND PRIVACY-PRESERVING TECHNIQUES IN FL



Security and Privacy Landscape of Federated Learning
Attacks and Threat
Attacks targeting security of the model
Data poisoning attack
Clean-label poisoning attack 1
Dirty-label poisoning attack
Model poisoning attack 2
Free-rider attack
Attacks targeting privacy of the model
Model inversion attack 3
Inference attack
Membership inference attack
Category attack 4
Security Enhancing and Privacy-Preserving Mechanisms
Security Enhancing Mechanisms
Defenses against Data poisoning attack
Defenses against Clean-label poisoning 5
Defenses against Dirty-label poisoning 6
Defenses against model poisoning attack 7, 8
Defenses against Free-riders attack 9
Privacy-preserving Mechanisms
Secure Multi-Party Computation (MPC)
Without Secure Aggregation 10
With Secure Aggregation 11, 12, 13
Verifiable Secure Aggregation 14
Hybrid method 19, 20
Differential Privacy (DP) 15, 16
Homomorphic Encryption (HE) 17, 18

Ren-Yi Huang and Dumindu Samaraweera, Exploring Threats, Defenses, and Privacy-Preserving Techniques in Federated Learning, *Accepted for publication IEEE Computer*, 2023

THREATS, DEFENSES, AND PRIVACY-PRESERVING TECHNIQUES IN FL



Security and Privacy Landscape of Federated Learning
Attacks and Threat
Attacks targeting security of the model
Data poisoning attack
Clean-label poisoning attack 1
Dirty-label poisoning attack
Model poisoning attack 2
Free-rider attack
Attacks targeting privacy of the model
Model inversion attack 3
Inference attack
Membership inference attack
Category attack 4
Security Enhancing and Privacy-Preserving Mechanisms
Security Enhancing Mechanisms
Defenses against Data poisoning attack
Defenses against Clean-label poisoning 5
Defenses against Dirty-label poisoning 6
Defenses against model poisoning attack 7, 8
Defenses against Free-riders attack 9
Privacy-preserving Mechanisms
Secure Multi-Party Computation (MPC)
Without Secure Aggregation 10
With Secure Aggregation 11, 12, 13
Verifiable Secure Aggregation 14
Differential Privacy (DP) 15, 16
Hybrid method 19, 20
Homomorphic Encryption (HE) 17, 18

Ren-Yi Huang and Dumindu Samaraweera, Exploring Threats, Defenses, and Privacy-Preserving Techniques in Federated Learning, *Accepted for publication IEEE Computer*, 2023

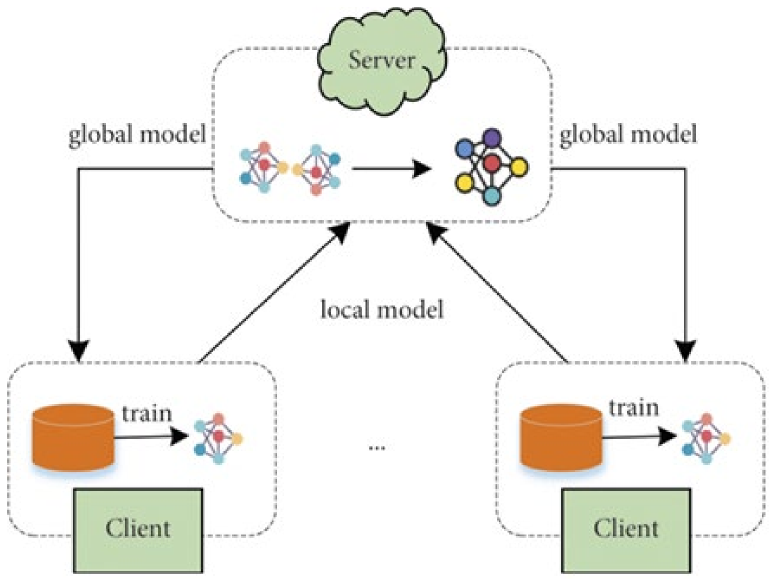
MODEL INVERSION ATTACK IN FL

Model inversion attacks can occur when an attacker tries to reconstruct a user's private data by analyzing the model parameters that were trained on that user's data.

- Goal: Reconstruct images using gradients from trained deep networks.

Assumptions

- Honest-but-curious server
- The server knows the number of images in training process
- The server is able to store, and process updated weights transmitted from each user



MODEL INVERSION ATTACK IN FL

1. The aggregated model is fixed after trained by clients

2. Randomly initialize the reconstructed data, and "train" the data with the objective function

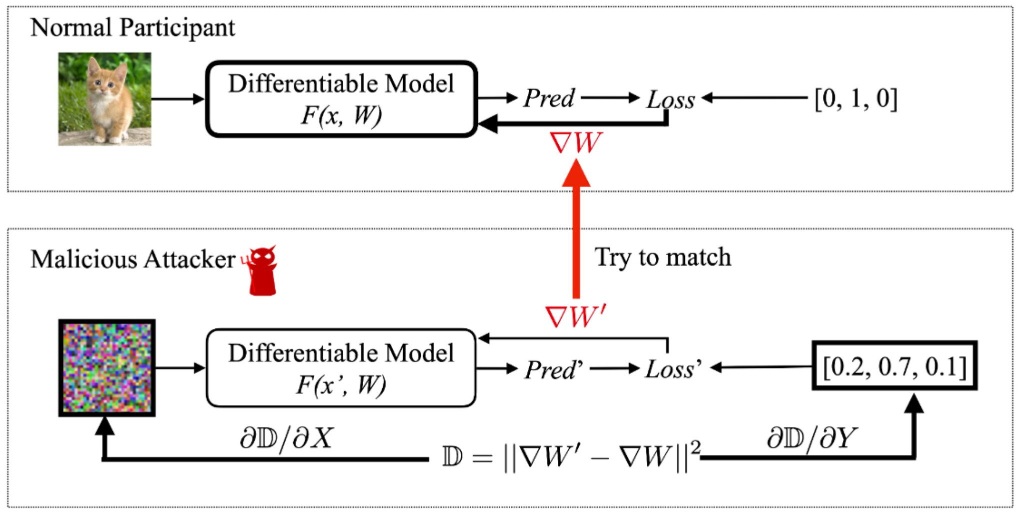
3. Calculate the difference of gradients between the reconstructed image and target images

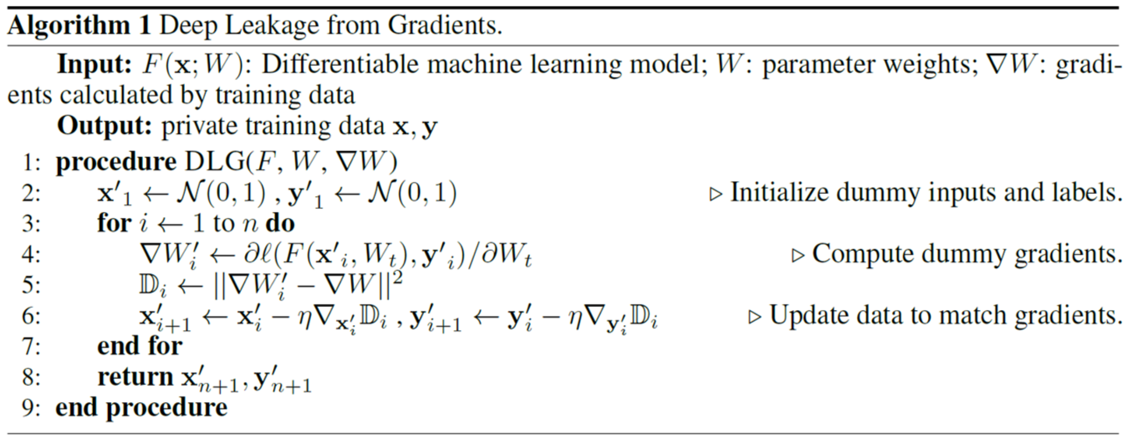
Euclidean cost function: focus on magnitude

$$\arg\min_{x} ||\nabla_\theta \mathcal{L}_\theta(x, y) - \nabla_\theta \mathcal{L}_\theta(x^*, y)||^2$$

Euclidean distance: $||\nabla W' - \nabla W||^2$.

Deep leakage by gradient matching




Algorithm 1 Deep Leakage from Gradients.

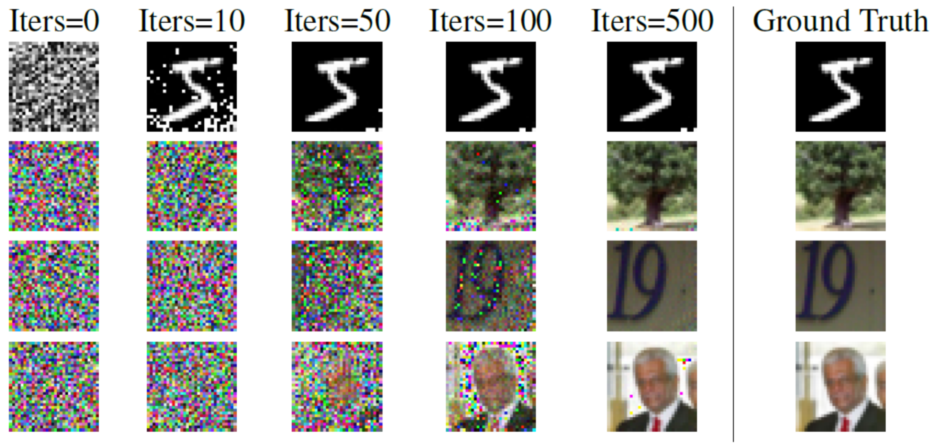
Input: $F(\mathbf{x}; W)$: Differentiable machine learning model; W: parameter weights; ∇W: gradients calculated by training data
Output: private training data $\mathbf{x}, \mathbf{y}$

```
1: procedure DLG($F$, $W$, $\nabla W$)
2:     $\mathbf{x}'_1 \leftarrow \mathcal{N}(0,1)$ , $\mathbf{y}'_1 \leftarrow \mathcal{N}(0,1)$        ▷ Initialize dummy inputs and labels.
3:     for $i \leftarrow 1$ to $n$ do
4:         $\nabla W'_i \leftarrow \partial \ell(F(\mathbf{x}'_i, W_t), \mathbf{y}'_i)/\partial W_t$        ▷ Compute dummy gradients.
5:         $\mathbb{D}_i \leftarrow ||\nabla W'_i - \nabla W||^2$
6:         $\mathbf{x}'_{i+1} \leftarrow \mathbf{x}'_i - \eta \nabla_{\mathbf{x}'_i} \mathbb{D}_i$ , $\mathbf{y}'_{i+1} \leftarrow \mathbf{y}'_i - \eta \nabla_{\mathbf{y}'_i} \mathbb{D}_i$        ▷ Update data to match gradients.
7:     end for
8:     return $\mathbf{x}'_{n+1}, \mathbf{y}'_{n+1}$
9: end procedure
```

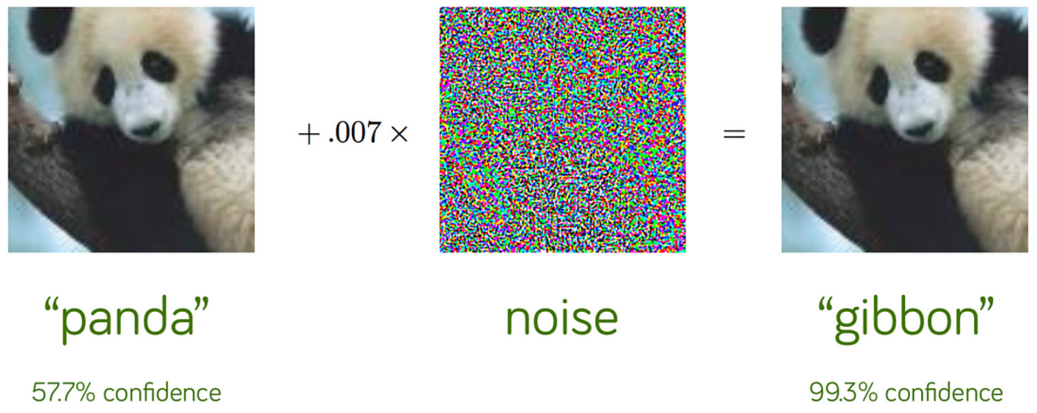
MODEL INVERSION ATTACK - EXPERIMENTAL RESULTS

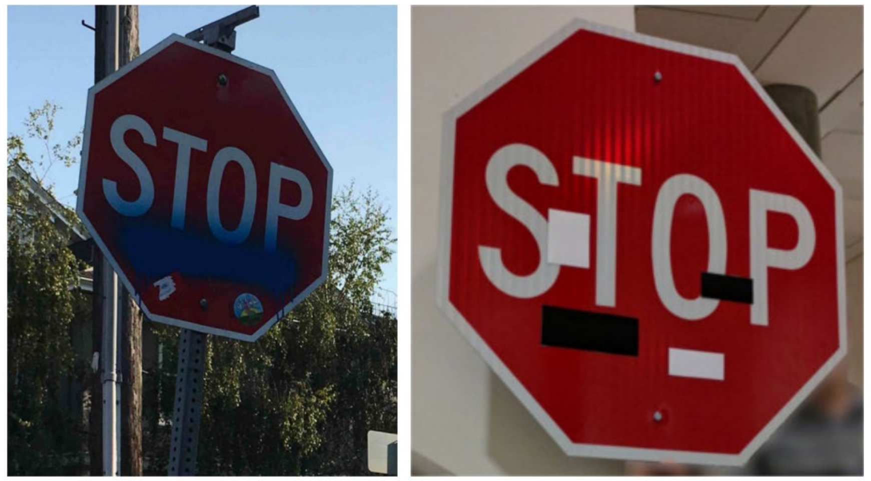
Results from DLG (Deep Leakage Gradients) attack (use Euclidean distance as the objective function).

Images with a clean background (For example MNIST) are easiest to recover, while complex images like face take more iterations to recover.



MODEL POISONING WITH ADVERSARIAL IMAGES





- Image of a panda, which the neural network correctly recognizes as a "panda" with 57.7% confidence.
- Add a little bit of carefully constructed noise and the same neural network now thinks this is an image of a gibbon with 99.3% confidence.

- The left image shows real graffiti on a Stop sign, something that most humans would not think is suspicious.
- The right image shows a physical perturbation applied to a Stop sign. The systems classify the sign on the right as a Speed Limit: 45 mph sign (completely different class).

HOW TO PROTECT DATA PRIVACY IN FL

There are three main approaches to protect privacy of the data in FL:

- Secure Multi-party Computation (MPC)
  - Any client can not see others' data
- Differential Privacy
  - Only share part of the data without revealing the sensitive one by adding noise
- Homomorphic Encryption
  - Encrypt the data before sending to the server

USING DIFFERENTIAL PRIVACY (DP)

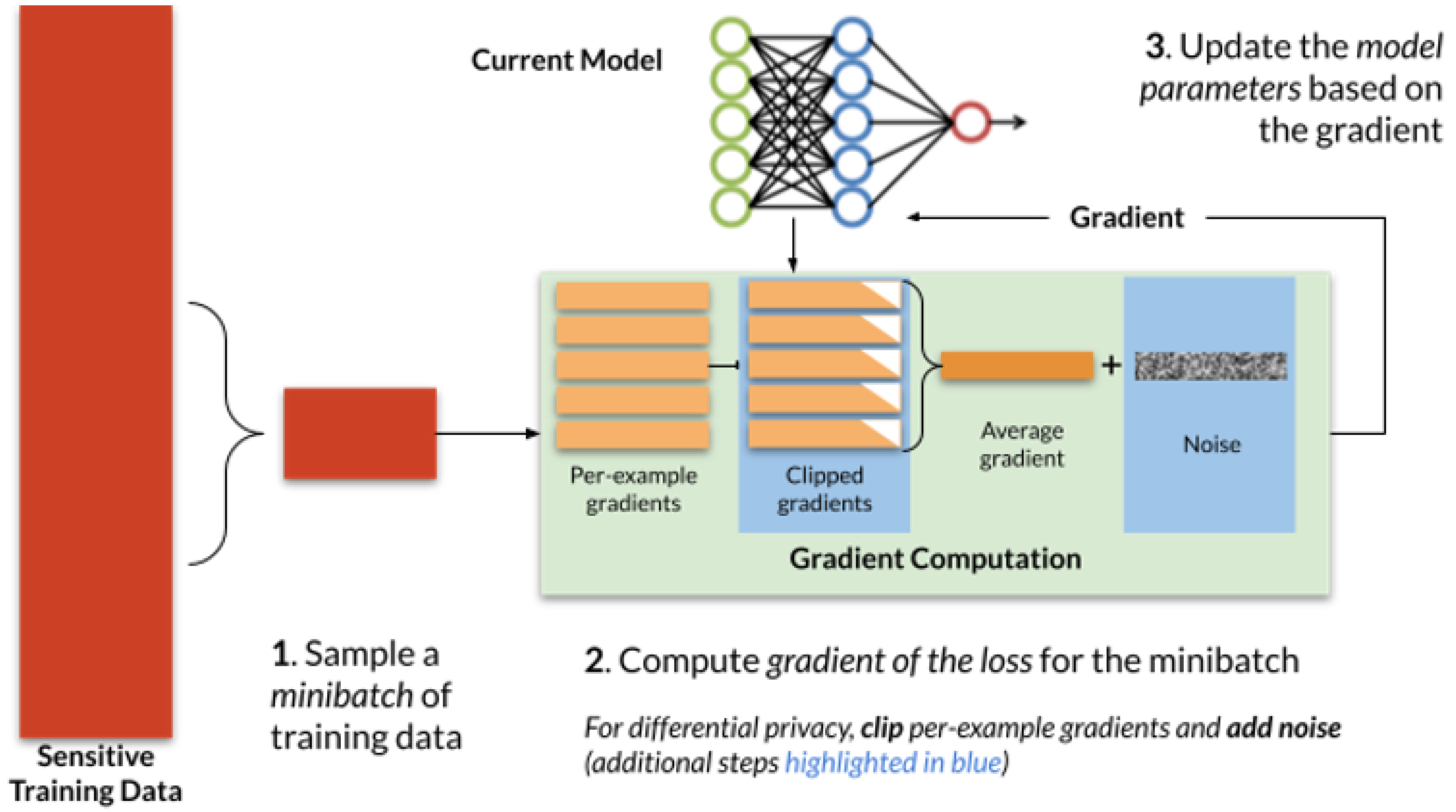
The core idea behind differential privacy is to add carefully calibrated noise or randomness to the data before it is released.

This noise ensures that the statistical properties of the dataset remain largely intact while making it extremely difficult to identify or infer information about any specific individual in the dataset.

Challenge:

- Even though DP can ensure strong information theoretic guarantees and also can be easily implemented in FL framework, it also leads to certain accuracy drops for practical applications.
- Its lossy!

USING DIFFERENTIAL PRIVACY (DP)


Current Model
3. Update the *model parameters* based on the gradient
Gradient
Per-example gradients
Clipped gradients
Average gradient
+
Noise
Gradient Computation
Sensitive Training Data
1. Sample a *minibatch* of training data
2. Compute *gradient of the loss* for the minibatch
*For differential privacy, **clip** per-example gradients and **add noise** (additional steps highlighted in blue)*

Differentially Private Stochastic Gradient Descent (DP-SGD)

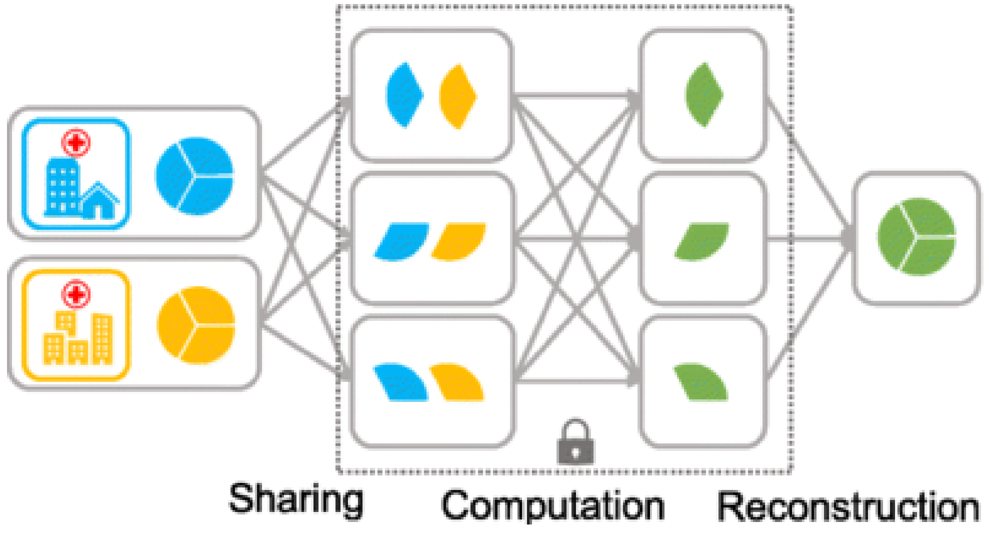
Image: https://www.nist.gov/blogs/cybersecurity-insights/how-deploy-machine-learning-differential-privacy

SECURE MULTI-PARTY COMPUTATION (MPC)

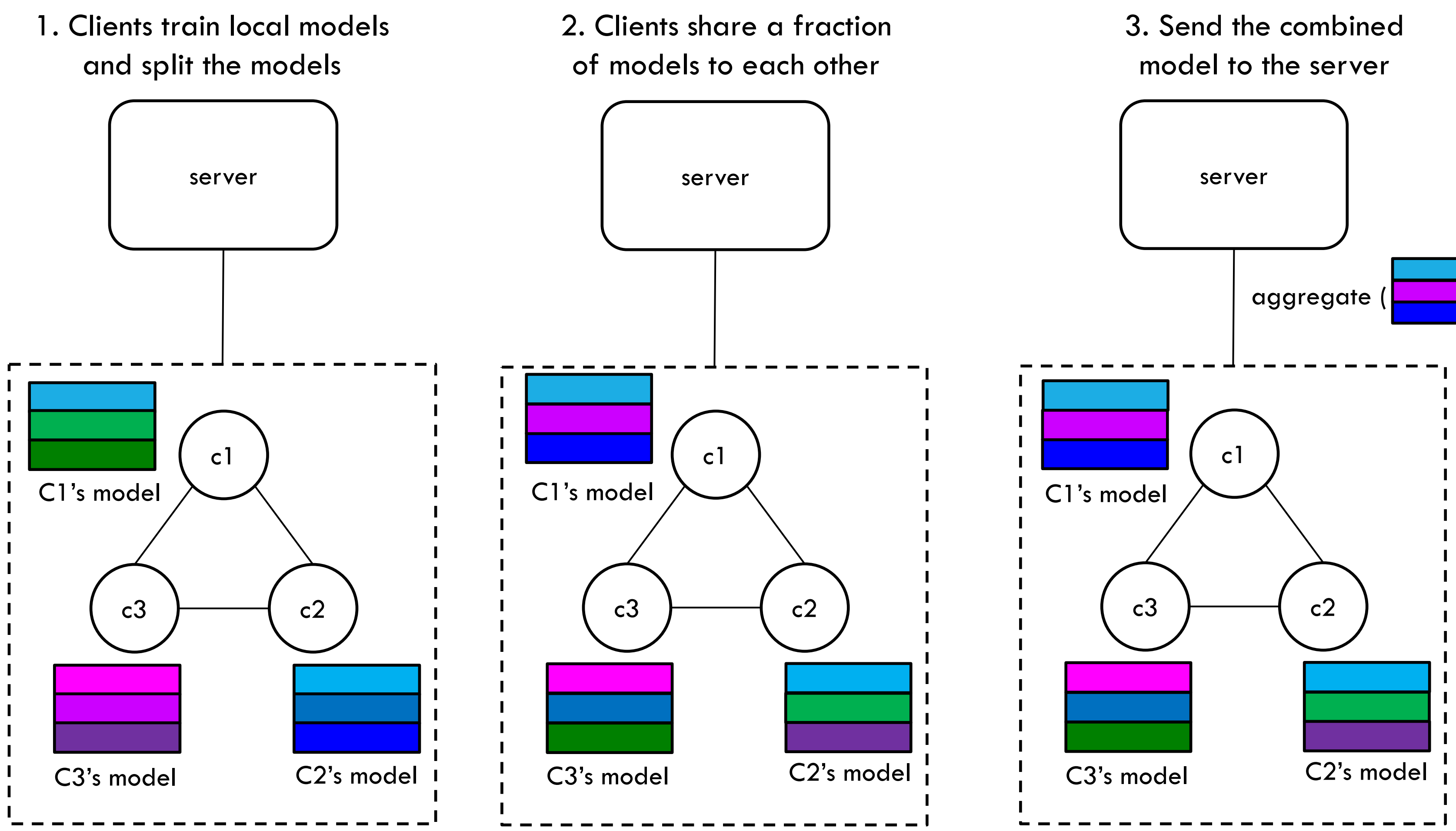
Overview:

MPC is a cryptographic technique that enables multiple parties to jointly compute a function over their respective private inputs while keeping those inputs confidential.

In essence, MPC allows parties to collaborate on a computation without revealing their sensitive data to each other.



MPC WITH FEDERATED LEARNING


1. Clients train local models and split the models
2. Clients share a fraction of models to each other
3. Send the combined model to the server
server
server
server
aggregate ( )
c1
c2
c3
C1's model
C2's model
C3's model

Weight parameters of the global model at the t-th iteration, which is the start point of next round of local training

$$w^{t+1} = \frac{1}{m}\sum_{i=1}^{m} w_i^t$$

Weight parameters of the i-th participant on the t-th iteration

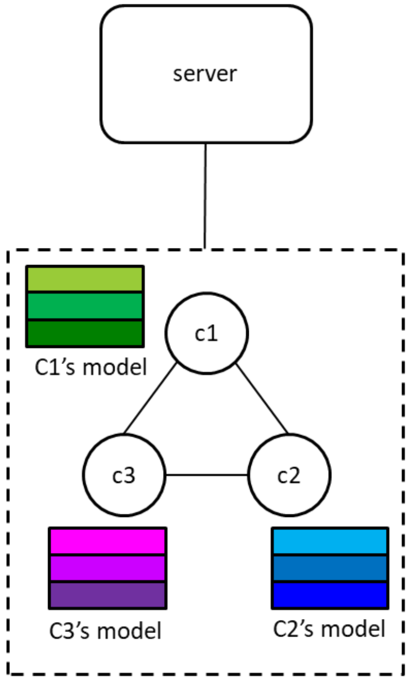
We have m local participants

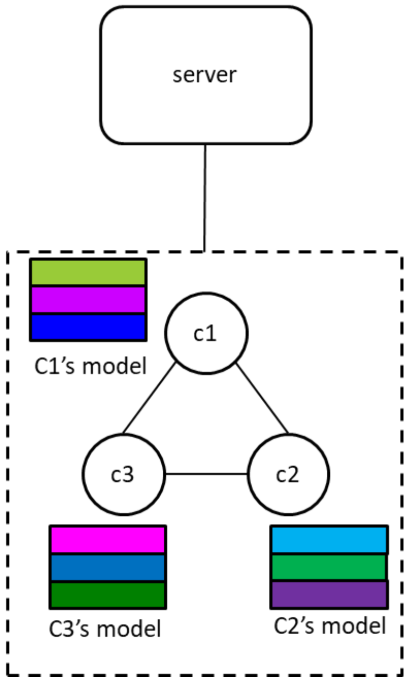
FIRST ROUND MODEL DECOMPOSITION

- Split the model into secret shares
- Transmit secret shares to neighbors

$$w_i^t = \sum_{j=1}^{m} w_i^{j,t} \quad \text{for} \quad i \in [1, 2, \cdots, m].$$

- i: the i-th client
- j: the j-th client
- t: the t-th training round
- $m_i^{j,t}$: a model piece transmitted from client i to client j on the t-th round
- m_i^t: client i's model on the t-th round





- The idea of this model decomposition is that rather than having a single model handled by a single participant, it is generally more secure to decompose the model to multiple parties.
- Basically, a single fraction does not present any useful information.

SECOND ROUND MODEL DECOMPOSITION

- After transmitting secret shares to neighbors, received model shares will then go though a second round of decomposition to create public pieces and private pieces:

$$w_j^{i,t} = \tilde{w}_j^{i,t} \oplus \bar{w}_j^{i,t}$$

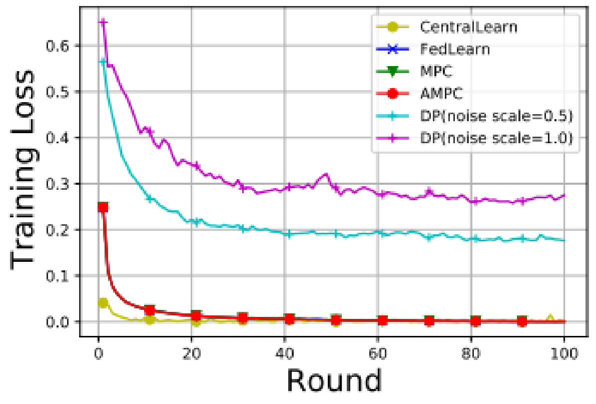
Public piece to be sent to the server

Random bias term as private model piece

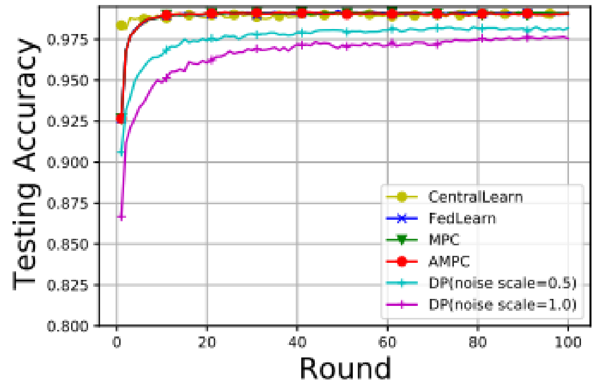
- The decomposition $\oplus$ is: simply generates a random local share and then deduct it from the true model.
- Key idea here is all neighbors receive the same bias terms from client i, so that each client can remove the bias term after receiving the aggregated model from the server.

EXPERIMENTAL RESULTS

- Datasets: MNIST, CIFAR-10 (random split)
- Network: LeNet-5 and VGG-16
  - Optimizer : Adam
  - Batch size: 128
  - Learning rate: 10^{-3}
  - 10 clients and 1 server
  - Clients' local epoch: 5
- Compared approaches:
  - Central Learn: centralized learning in the server only
  - Fed Learn: without MPC or DP
  - MPC: standard MPC algorithm from PySyft
  - DP: differential privacy
  - AMPC: proposed method (Augmented MPC)
- Accuracy: classification accuracy

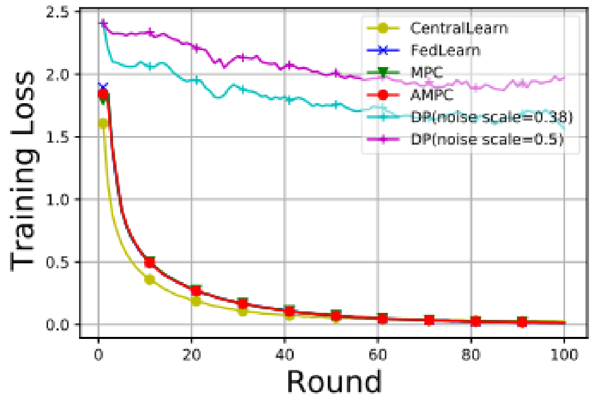


(a) Training loss on the MNIST dataset.

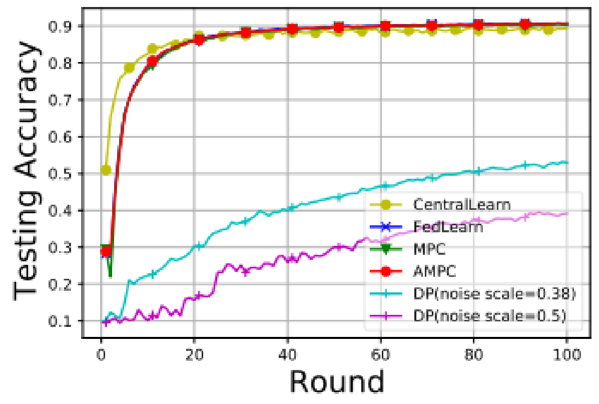


(b) Testing accuracy on the MNIST dataset.

Fig. 2: Performance of LeNet-5 on the MNIST dataset. Samples are randomly distributed to 10 participants.



(a) Training loss on the CIFAR-10 dataset.



(b) Testing accuracy on the CIFAR-10 dataset.

Fig. 3: Performance of VGG-16 on the CIFAR-10 dataset. Samples are randomly distributed to 10 participants.

SUMMARY

Overall, it is important to design federated learning systems with privacy in mind and to use a combination of different techniques to protect against privacy attacks such as model inversion.

- By doing so, it is possible to enable collaborative machine learning while preserving the privacy of individual users.

Multi step decomposition for secret shares in MPC algorithms, can ensure true global model is invisible to the central server.

THANK YOU!

HOW SHAMIR'S SECRET SHARING WORKS

Suppose we need to share a value $s = 1954$ among $n = 4$ participants with threshold $t = 3$.

Shamir sharing scheme has a total number of shares (n) and a threshold (t).

- The threshold is the number of shares required to reconstruct the original secret.
- For example, with five shares and a threshold of three, you only need three of the five shares to calculate the original secret.

Step 1: Randomly choose $t - 1$ integers, say 43 and 12.

- Then we build a polynomial in the form $y = a0 + a1 * x + a2 * x^2$ where $a0$ is the secret and $a1$ and $a2$ are the randomly chosen integers.

Step 2: Distribute the shares among participants.

- Share 1: Where $x = 1$ and $y = 2009$
- Share 2: Where $x = 2$ and $y = 2088$
- Share 3: Where $x = 3$ and $y = 2191$
- Share 4: Where $x = 4$ and $y = 2318$

Step 3: Reconstruct.

- Suppose we have 3 shares (1, 2009), (2, 2088) and (4, 2318). Since our threshold is 3. we can define a parabola with these 3 points and calculate $a0$, the original value.