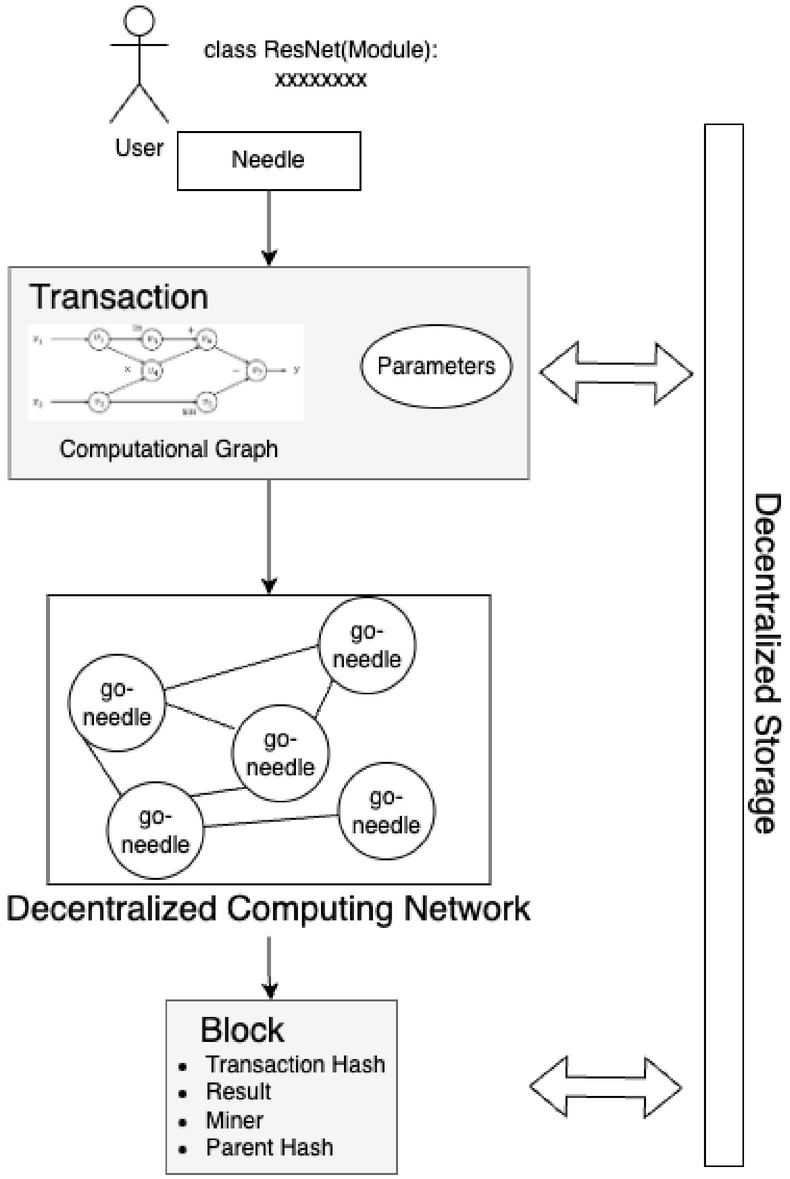
Security of the decentralized ML scheme

Decentralized ML scheme

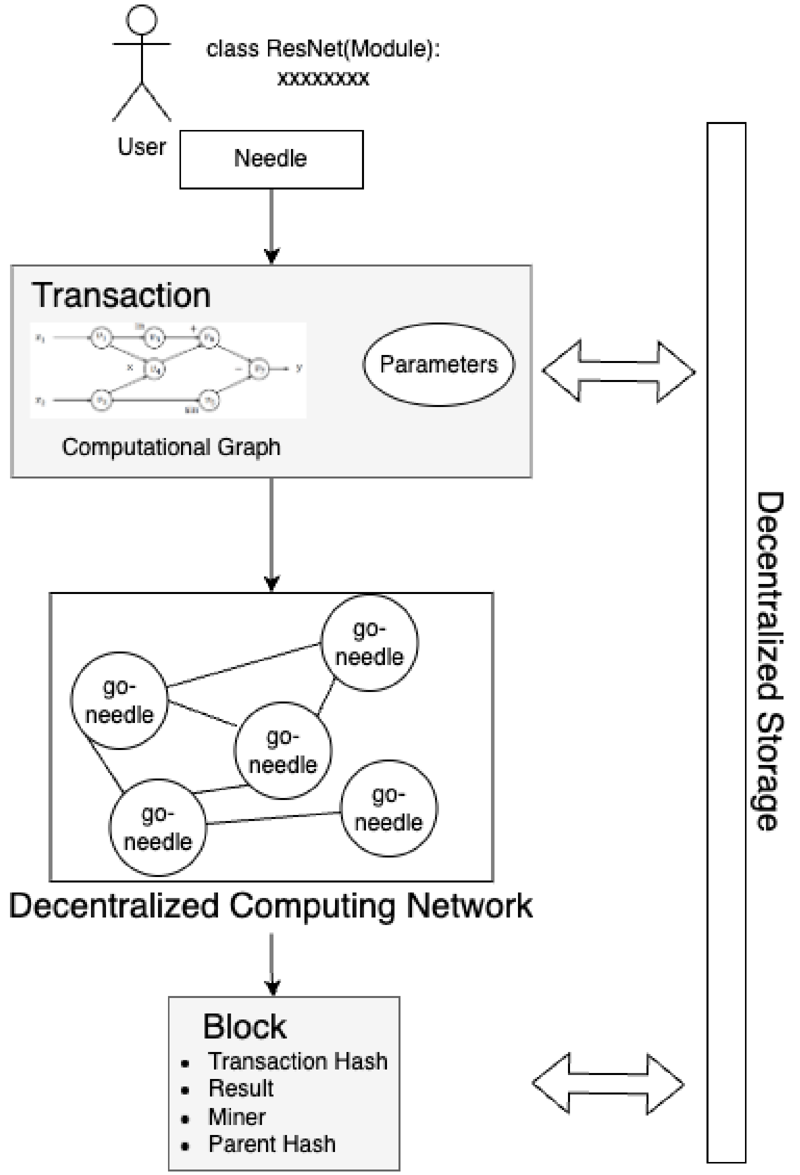
- user sends a computing task to network
- we assume data available in a distributed store
    - we also assume public for now
- network distributes task among miner/verifier nodes
- miners computes part of task, stores result in a block
- verifier retrieves and verifies block, on success recieves reward



Threat model for this scheme

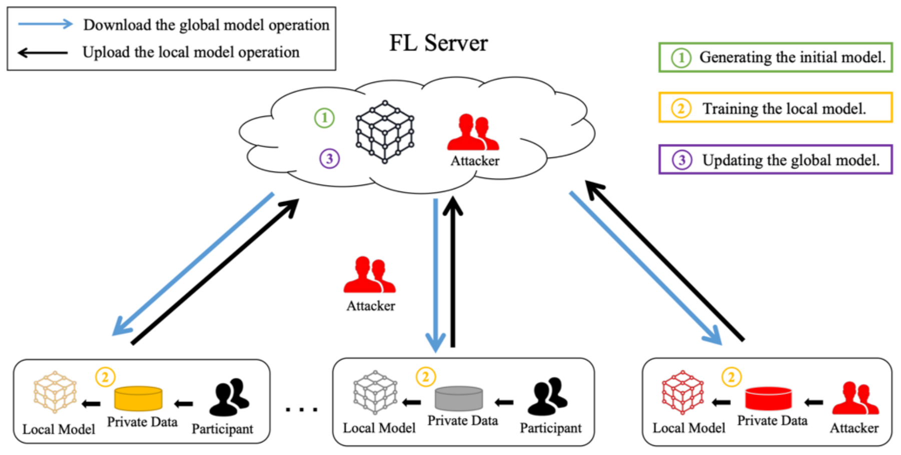
- Byzantine model updates
  - cheaper to not compute the task, send noise
  - or steal (rebroadcast) a previous result
- Model privacy
  - paying to compute the model
- Fair payment
  - ensure nodes get paid

Relevant areas:
- data marketplaces
- decentralized federated learning
- decentralized machine learning



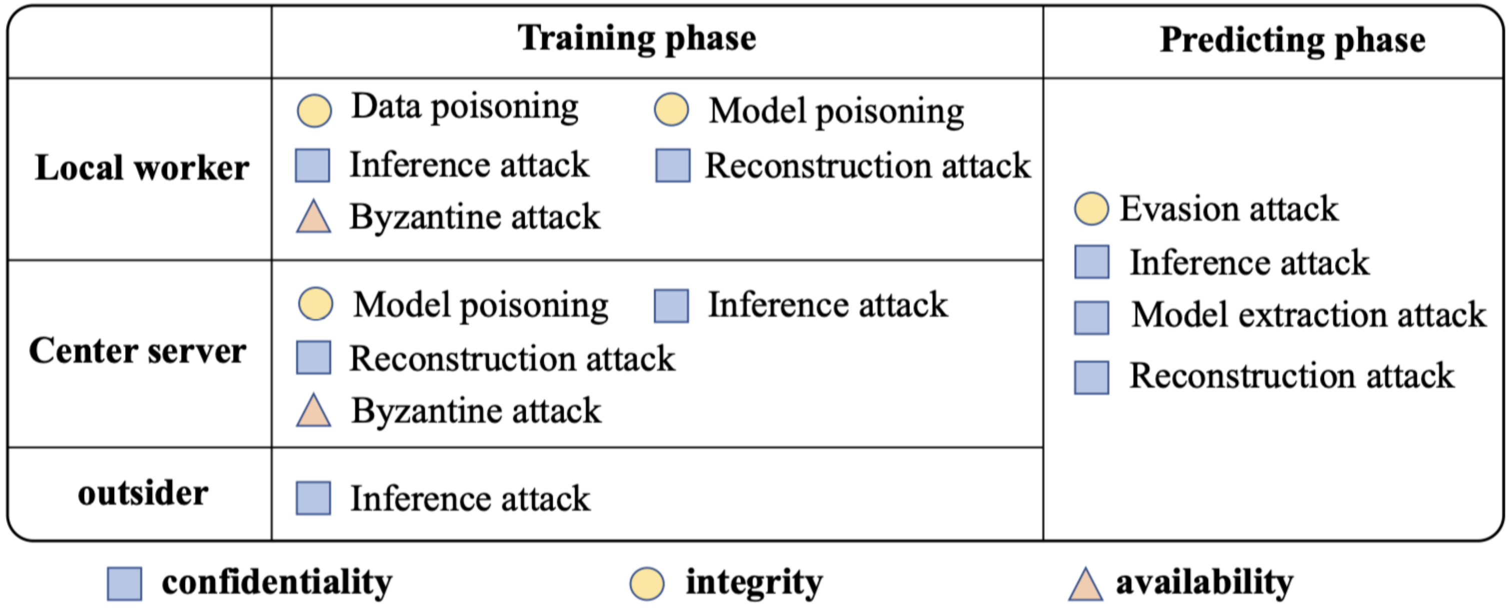
Federated learning [1]

Some relevant threat modelling
- byzantine attacks at local worker
- model poisoning (dismiss for now)
- model extraction attack



Download the global model operation
Upload the local model operation
FL Server
1 Generating the initial model.
2 Training the local model.
3 Updating the global model.
Attacker
Attacker
Local Model
Private Data
Participant
Local Model
Private Data
Participant
Local Model
Private Data
Attacker
FL participants

Fig. 2: Threats in a federated learning environment.



	Training phase	Predicting phase
Local worker	● Data poisoning ● Model poisoning ■ Inference attack ■ Reconstruction attack ▲ Byzantine attack	● Evasion attack ■ Inference attack ■ Model extraction attack ■ Reconstruction attack
Center server	● Model poisoning ■ Inference attack ■ Reconstruction attack ▲ Byzantine attack	
outsider	■ Inference attack	

■ **confidentiality** ● **integrity** ▲ **availability**

Major differences in setting
- model is not served
- rewarding nodes
- data availability

Threats considered
- byzantine model updates
- model extraction
- fair rewards for work

Chain FL (2020) [2]

- private blockchain
- chunk data onto the network
- smart contracts
- proof of authority
  - test dataset
- online federated averaging

- acknowledges problems with
  - byzantine updates
  - decentralized storage
- also global model shared

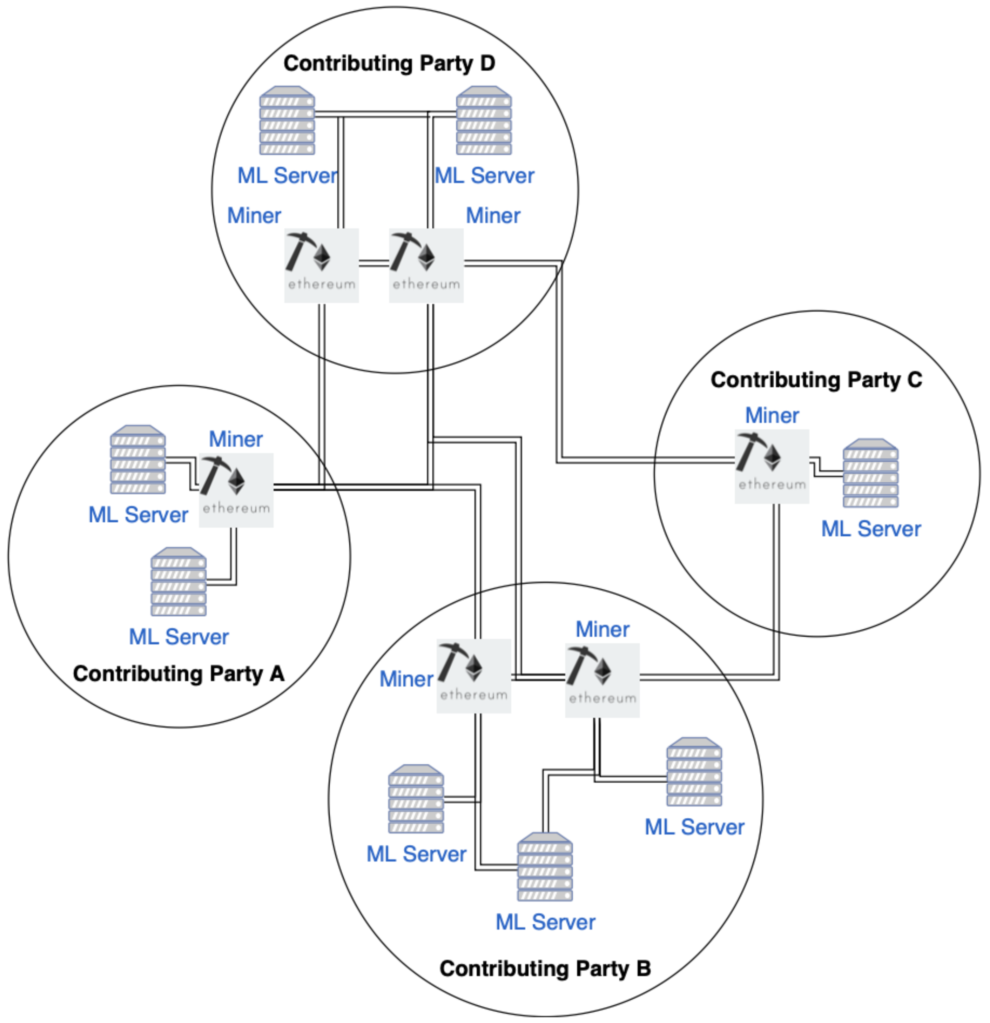


Fig. 1: Illustration of the sample system architecture with four contributing parties.

LearningChain (2018) [3]

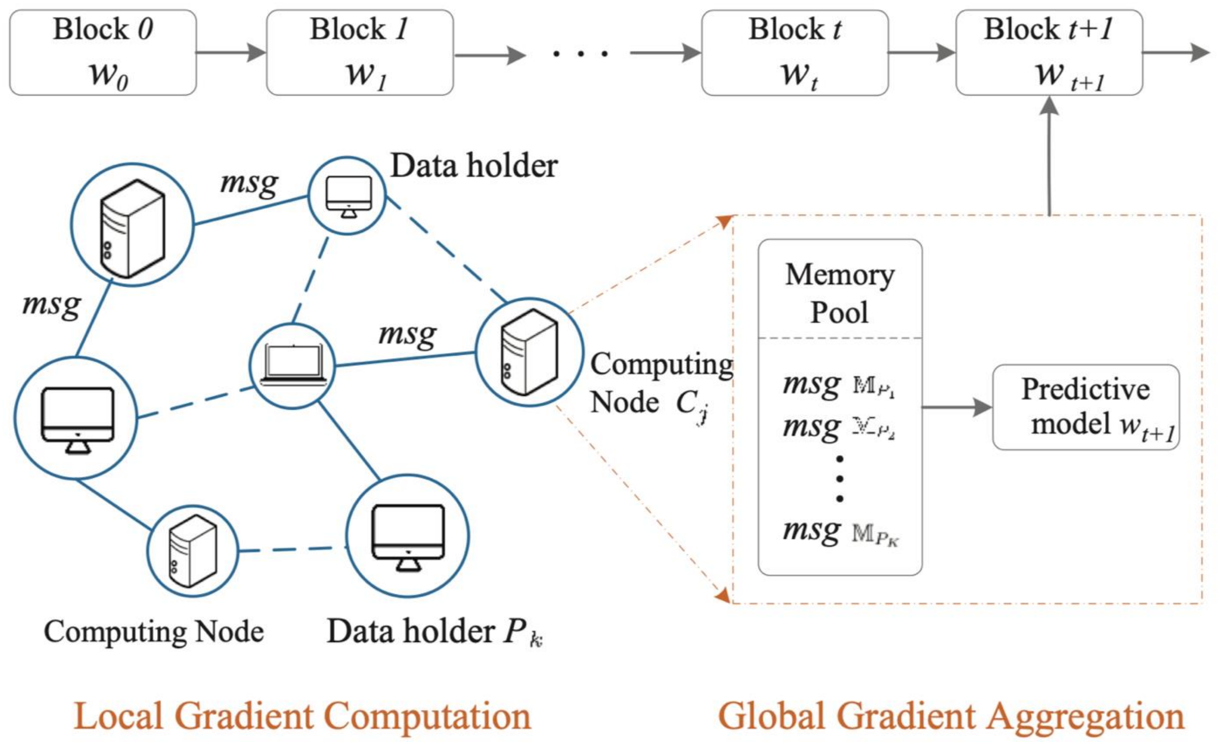


Fig. 4. The Training Process of LearningChain

1. blockchain initialization
    1. establish network
    2. chosen node broadcasts task
    3. reach consensus
2. local gradient computation
    1. data holder
        1. retrieve weights from latest block
        2. compute local gradient
        3. broadcast gradient (DP)
    2. computing node
        1. compete for proof-of-work
        2. compute global gradient
            1. l-nearest aggregation
        3. make block containing new weights
            1. backwards to last good block

Problems
- global model
- proof-of-work not tied to executing task well

Omnilytics (2021) [4]

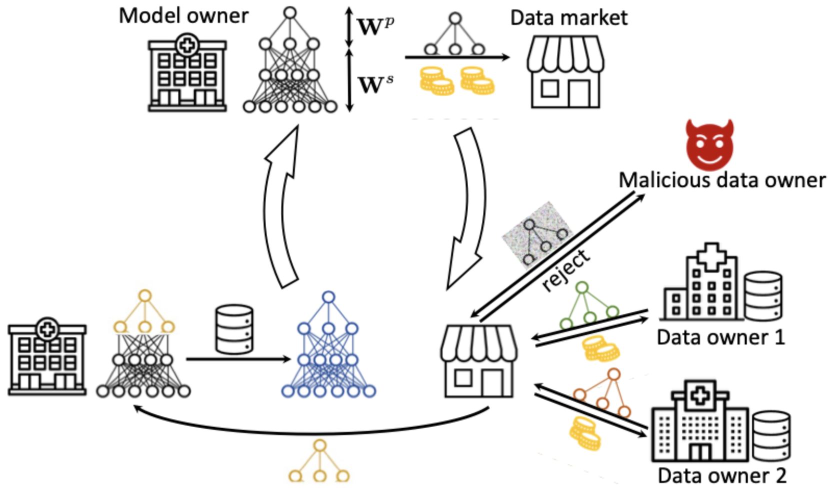


Figure 1: An overview of the operation of the proposed OmniLytics data market. The model owner splits its initial model into a private model W^s and a public model W^p, updates the public model through the data market using data owners' private data, and combines the updated public model with the old private model and performs local model adaptation with its private data. The smart contract implementing the data market reimburses the honest data owners who contribute to updating public model, and rejects erroneous results from malicious data owners.

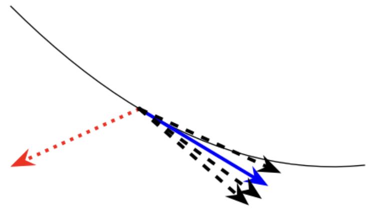


Figure 1: The gradient estimates computed by correct workers (black dashed arrows) are distributed around the actual gradient (solid arrow) of the cost function (thin black curve). A Byzantine worker can propose an arbitrary vector (red dotted arrow). [5]

- secure aggregation (MPC) for model updates [6]
- multi-Krum algorithm to reject bad updates [5]
    - vector closest to neighbours
    - squared distance

Threat model
- Model privacy
- Data privacy
- Resistance to Byzantine data owners
- Resistance to Byzantine model owner

Omnilytics in detail

- Combines solutions
  - private-public model splitting [7]
  - blockchain / smart contracts
  - secure aggregation / MPC
  - multi-Krum

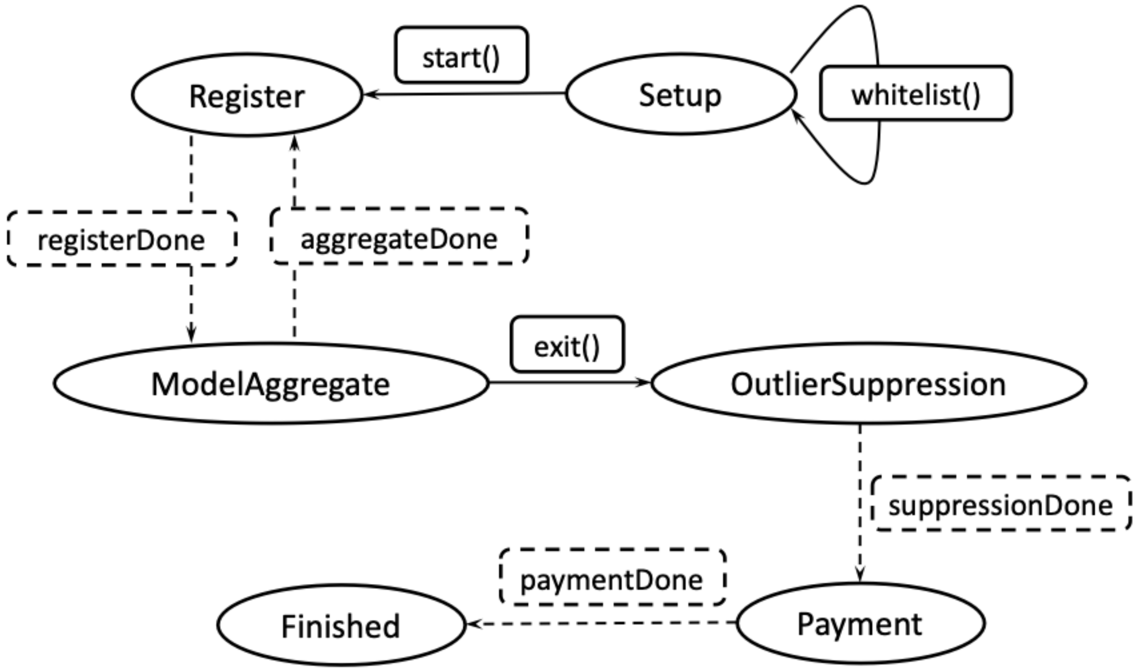


Figure 2: State transition of the smart contract SecModelUpdate. The six states are represented by ovals. State transitions are triggered by either applying a method (in a solid box), or occurrence of an event (in a dashed box).

Omnilytics more or less solves the problem

However…

- while public model is small, it could be shared many times
  - unconvinced that it solves model privacy generally, not evaluated [7]
  - eventually reconstruct much of the global model
  - especially when training from scratch
  - example only boosts MNIST from initial 60% to 80%
- needs IID data for multi-Krum
- tolerances of secure aggregation
  - but we don’t care about data privacy
- cost of updating global model

Data availability and sensitivity

Tradeoff between cost and security/privacy

- Locality will affect optimal node choice
- Additional redundancy might be necessary for general ML
    - and recomputation

If we also assume data is private, expands our threat model

- need to securely distribute
- data reconstruction attacks

Might want to ensure data can be safely shared (TEEs, at a cost)

Or homomorphic cryptography (but cost, support, and side-channels)

Next steps

- Verify that Omnilytics would work with proof-of-useful-work
- Implement a case study (MNIST)
- Test attacks for global model reconstruction from public model

- Analyze overhead
- Analyze costs VS security & privacy tradeoffs

- … Look at how to store data, be private
- … model poisoning defenses

References

1. Yao Chen, Yijie Gui, Hong Lin, Wensheng Gan, & Yongdong Wu. (2022). Federated Learning Attacks and Defenses: A Survey.
2. Korkmaz, C., Kocas, H., Uysal, A., Masry, A., Ozkasap, O., & Akgun, B. (2020). Chain FL: Decentralized Federated Machine Learning via Blockchain. In *2020 Second International Conference on Blockchain Computing and Applications (BCCA)* (pp. 140-146).
3. Chen, X., Ji, J., Luo, C., Liao, W., & Li, P. (2018). When Machine Learning Meets Blockchain: A Decentralized, Privacy-preserving and Secure Design. In *2018 IEEE International Conference on Big Data (Big Data)* (pp. 1178-1187).
4. Jiacheng Liang, Songze Li, Bochuan Cao, Wensi Jiang, & Chaoyang He. (2021). OmniLytics: A Blockchain-based Secure Data Market for Decentralized Machine Learning.
5. Blanchard, P., El Mhamdi, E., Guerraoui, R., & Stainer, J. (2017). Machine learning with adversaries: byzantine tolerant gradient descent. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (pp. 118–128). Curran Associates Inc..
6. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H., Patel, S., Ramage, D., Segal, A., & Seth, K. (2017). Practical Secure Aggregation for Privacy-Preserving Machine Learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (pp. 1175–1191). Association for Computing Machinery.
7. Paul Pu Liang, Terrance Liu, Ziyin Liu, Ruslan Salakhutdinov, & Louis-Philippe Morency (2020). Think Locally, Act Globally: Federated Learning with Local and Global Representations*. CoRR, abs/2001.01523.*