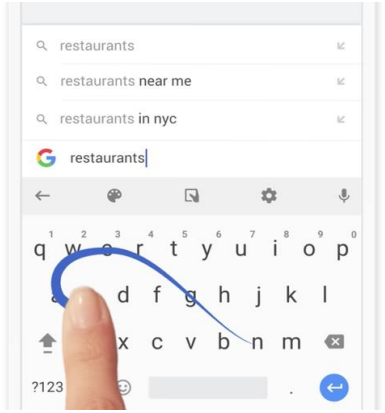
Do We Really Need to Design New Byzantine-robust Aggregation Rules?

Minghong Fang, Seyedsina Nabavirazavi, Zhuqing Liu, Wei Sun, Sundararaja Sitharama Iyengar, Haibo Yang

University of Louisville, Florida International University, University of North Texas, Wichita State University, Rochester Institute of Technology

NDSS 2025

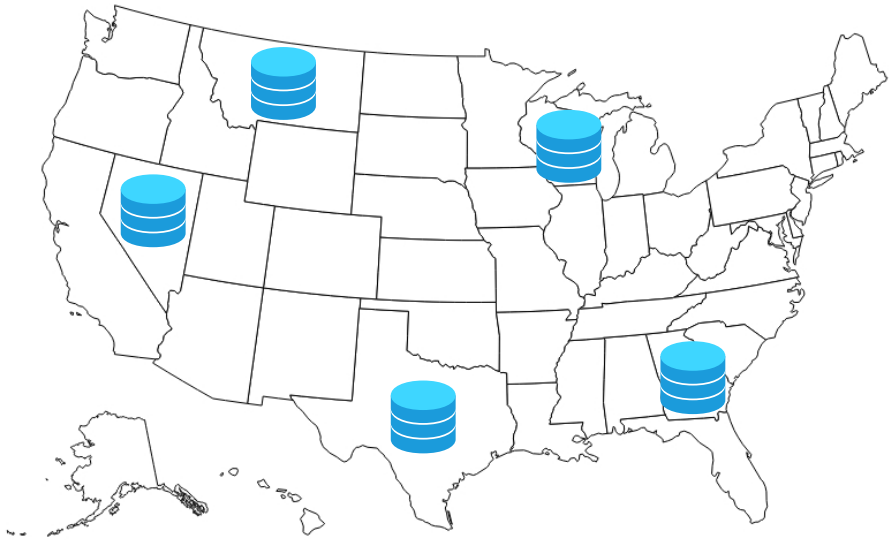
Machine Learning is Everywhere



...ranging from basic sensor automation to large-scale collaborative environments

Centralized Learning

Large scale data are typically distributed in different locations



Conventional approach: centralized learning

(All the data are moved to a single location)

Challenges of Centralized Learning

Challenges of Centralized Learning

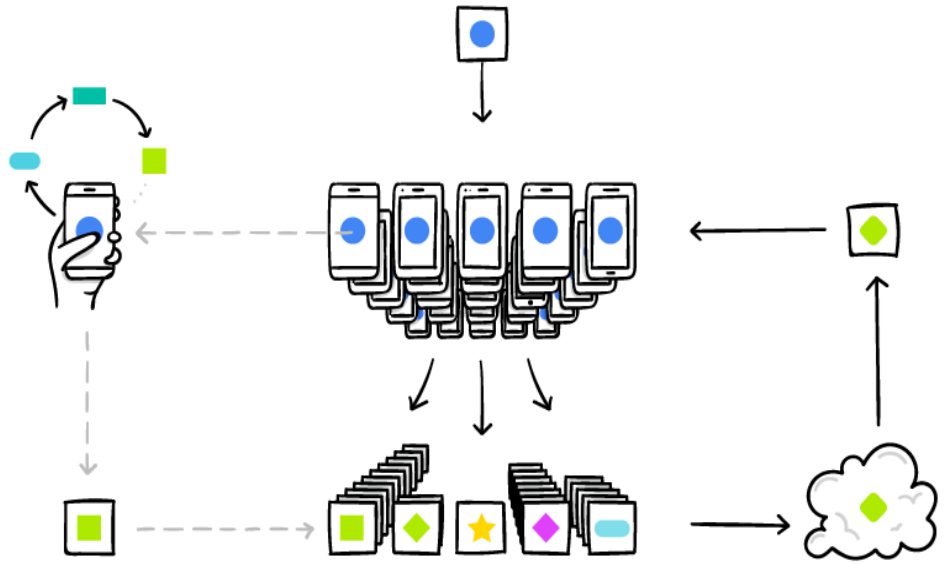
- Data leakage
  - Private data cannot be shared

Challenges of Centralized Learning

- Data leakage
  - Private data cannot be shared
- High communication cost
  - Intolerable for resource-constrained clients
    - Smartphone
    - IoT

Federated Learning

- Training data stay locally on clients
- Clients train models locally
- Clients send local model updates to the server
- Server aggregates the received model updates

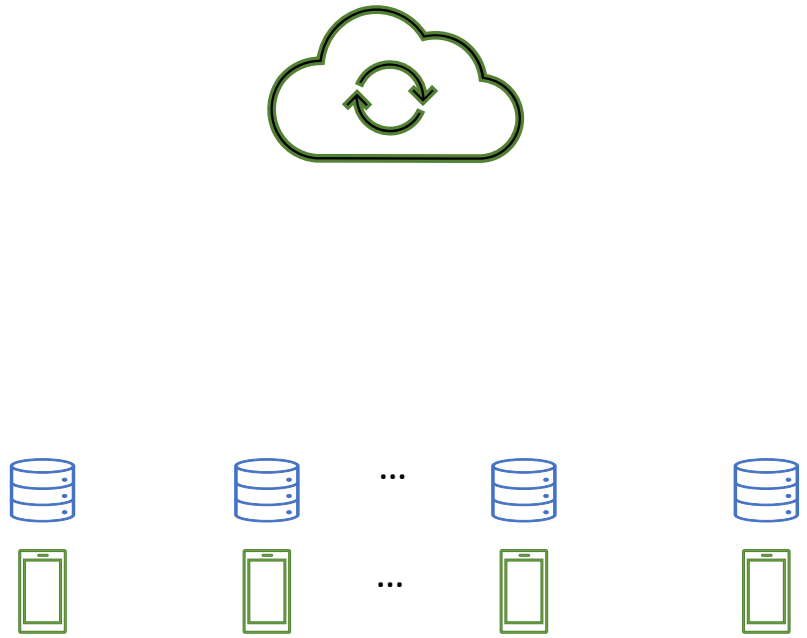


Federated Learning Background

Federated Learning Background

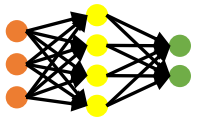
Federated Learning Background

Federated Learning Background

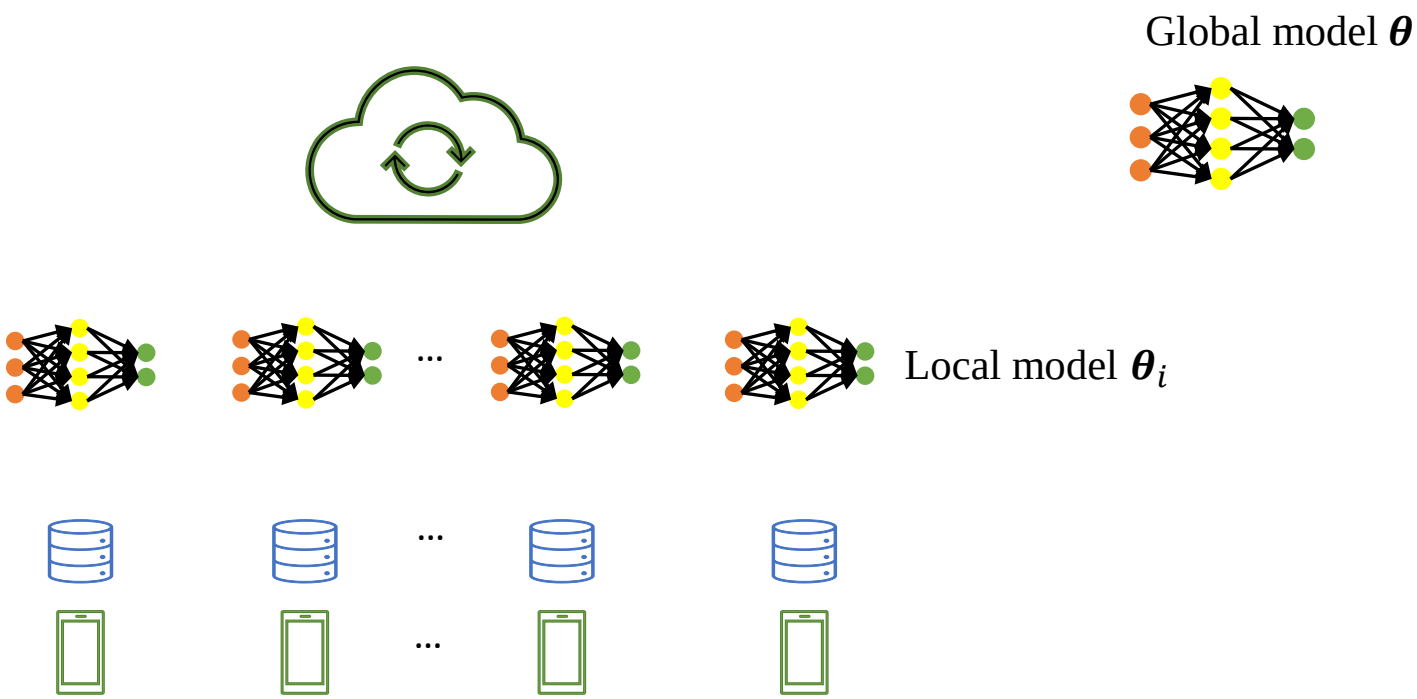


Federated Learning Background

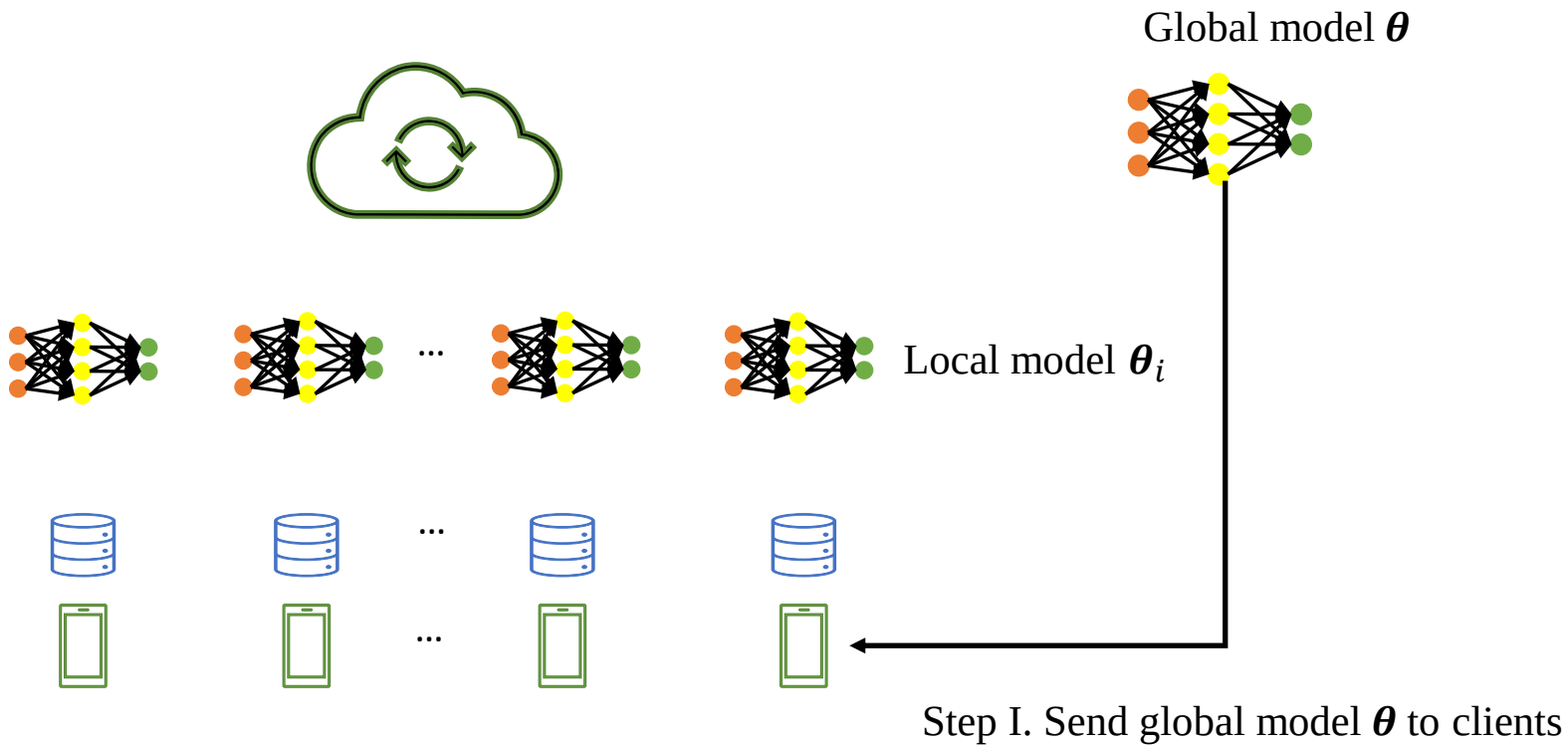
Global model $\boldsymbol{\theta}$



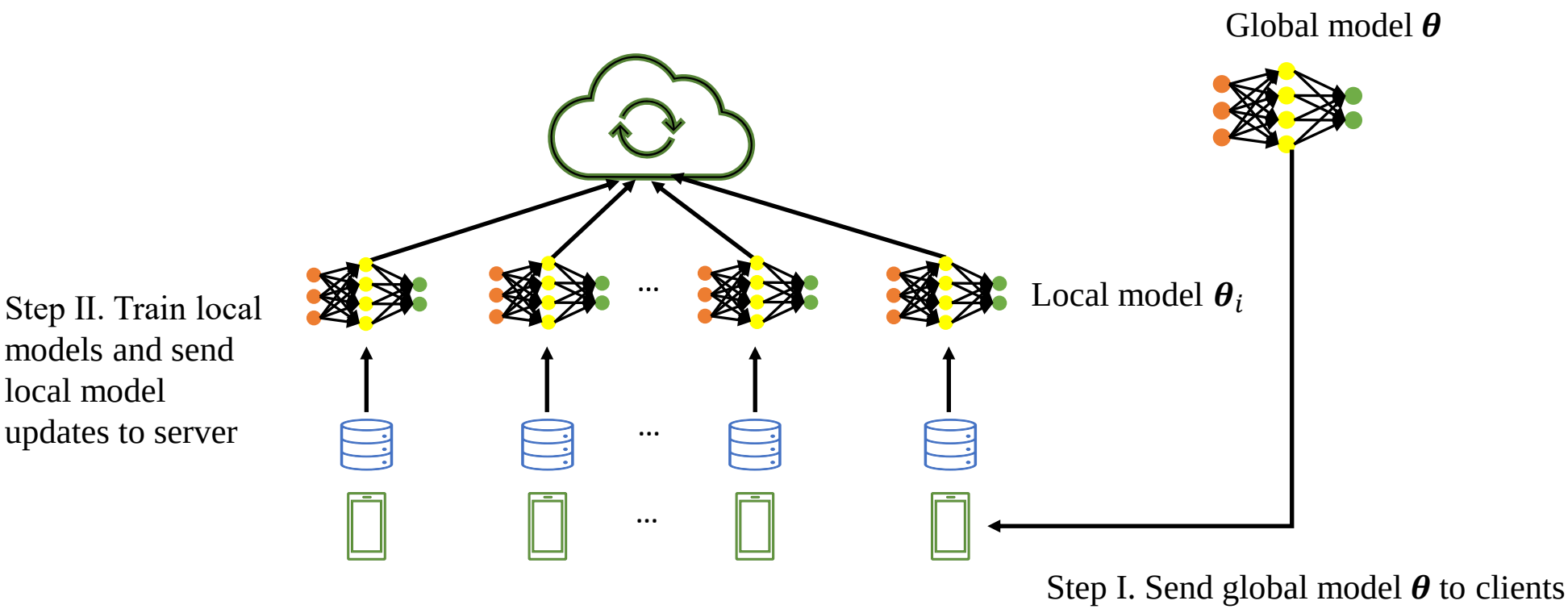
Federated Learning Background



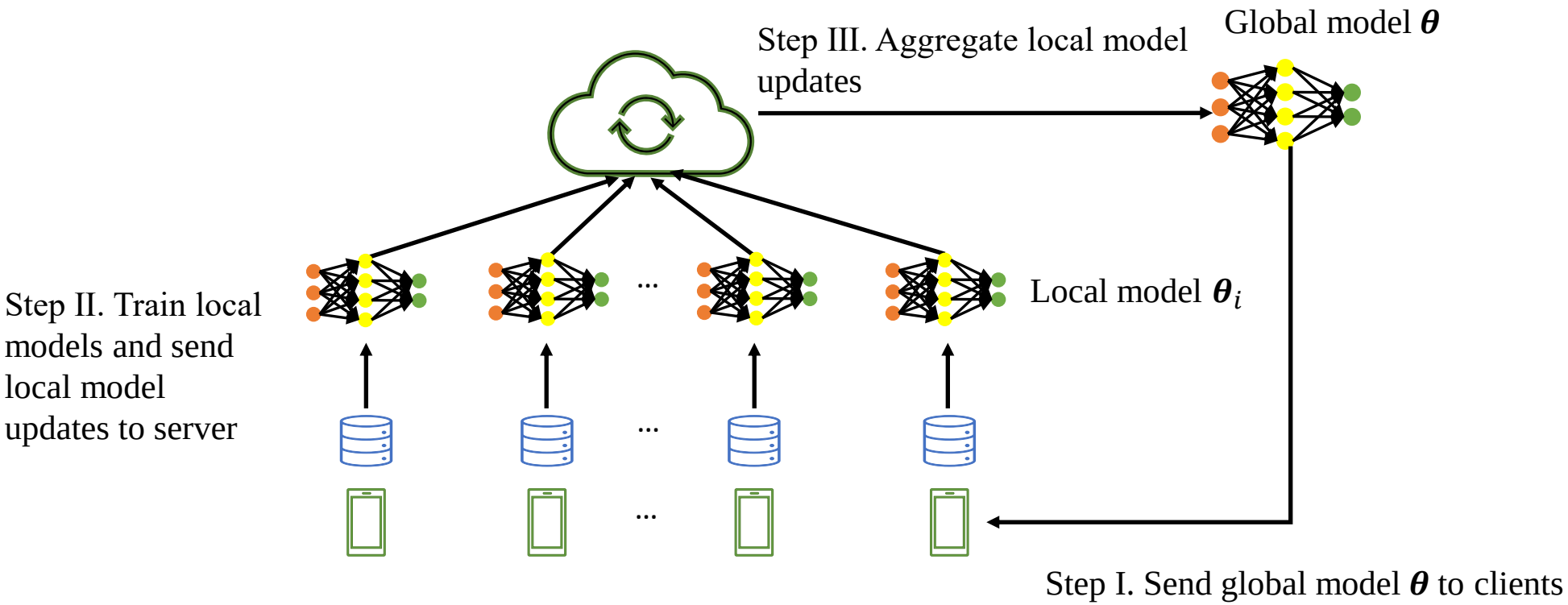
Federated Learning Background



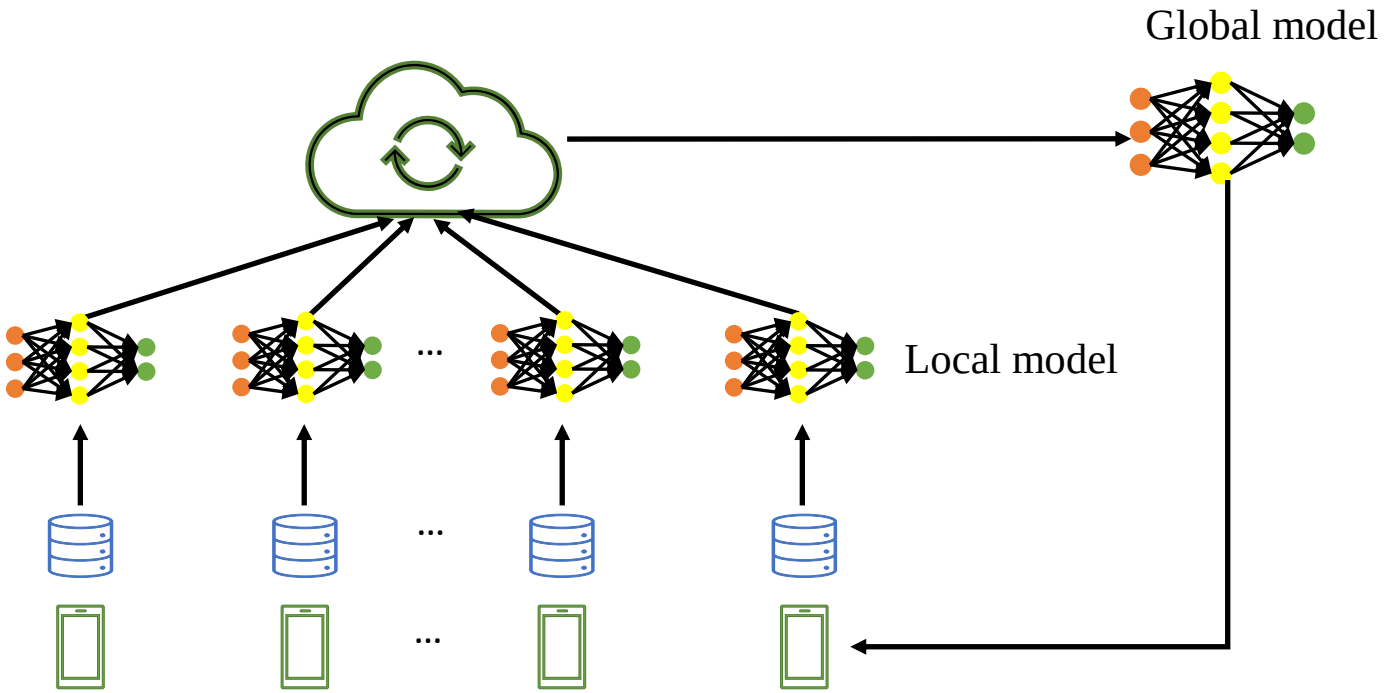
Federated Learning Background



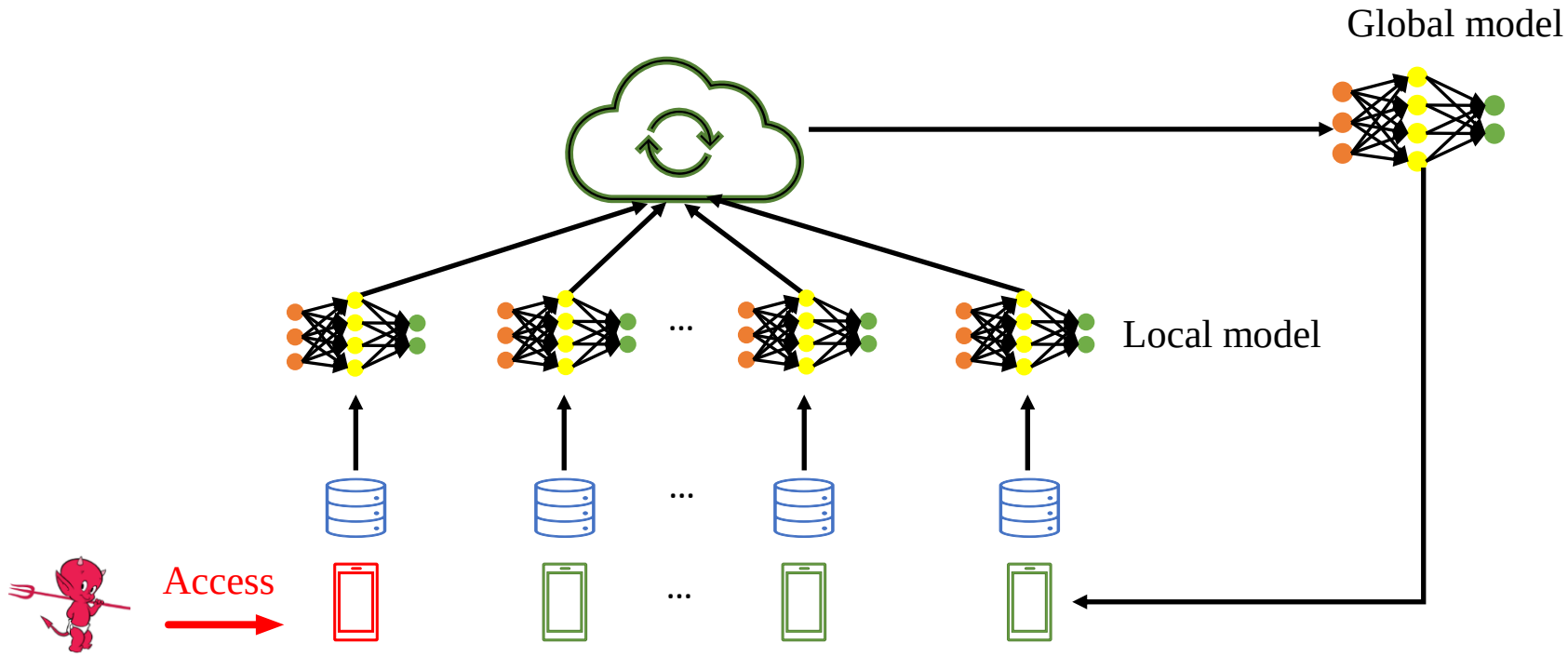
Federated Learning Background



Poisoning Attacks to Federated Learning

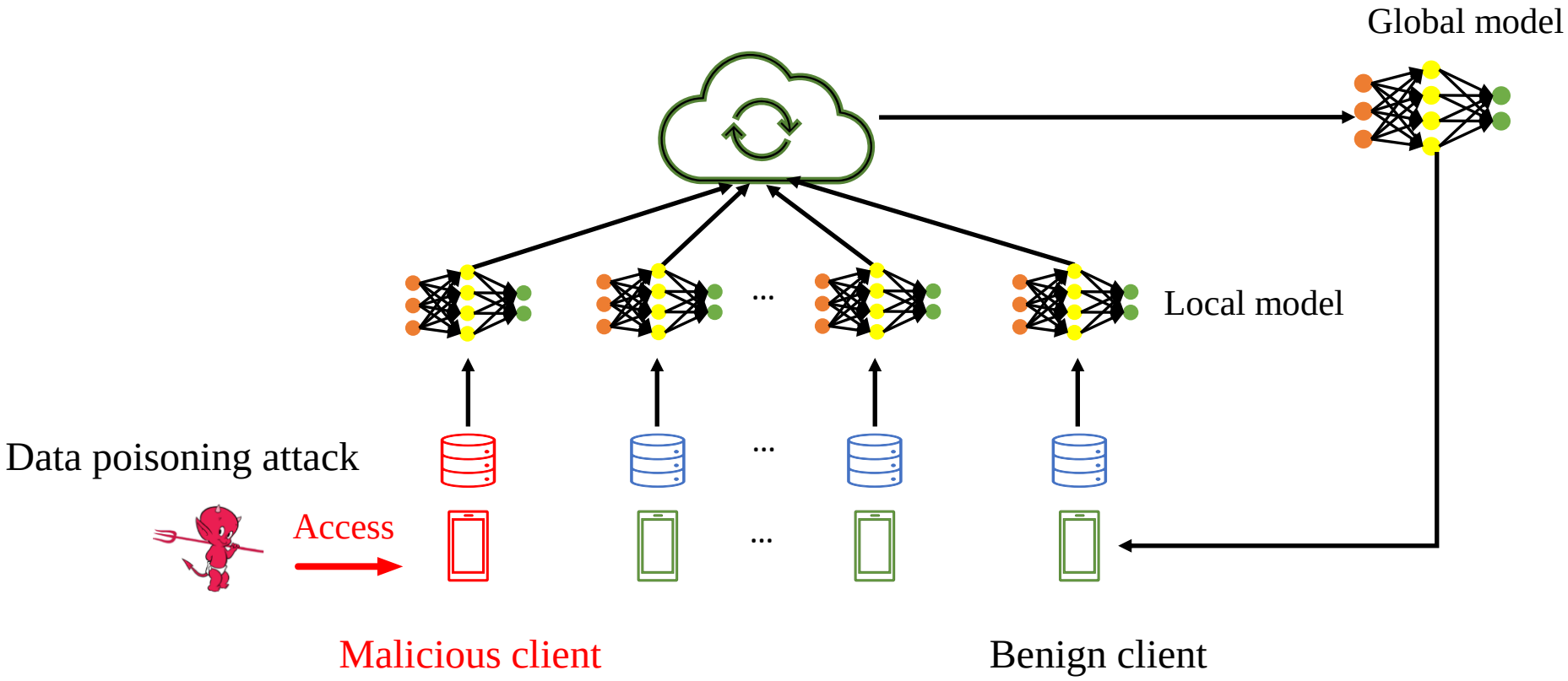


Poisoning Attacks to Federated Learning

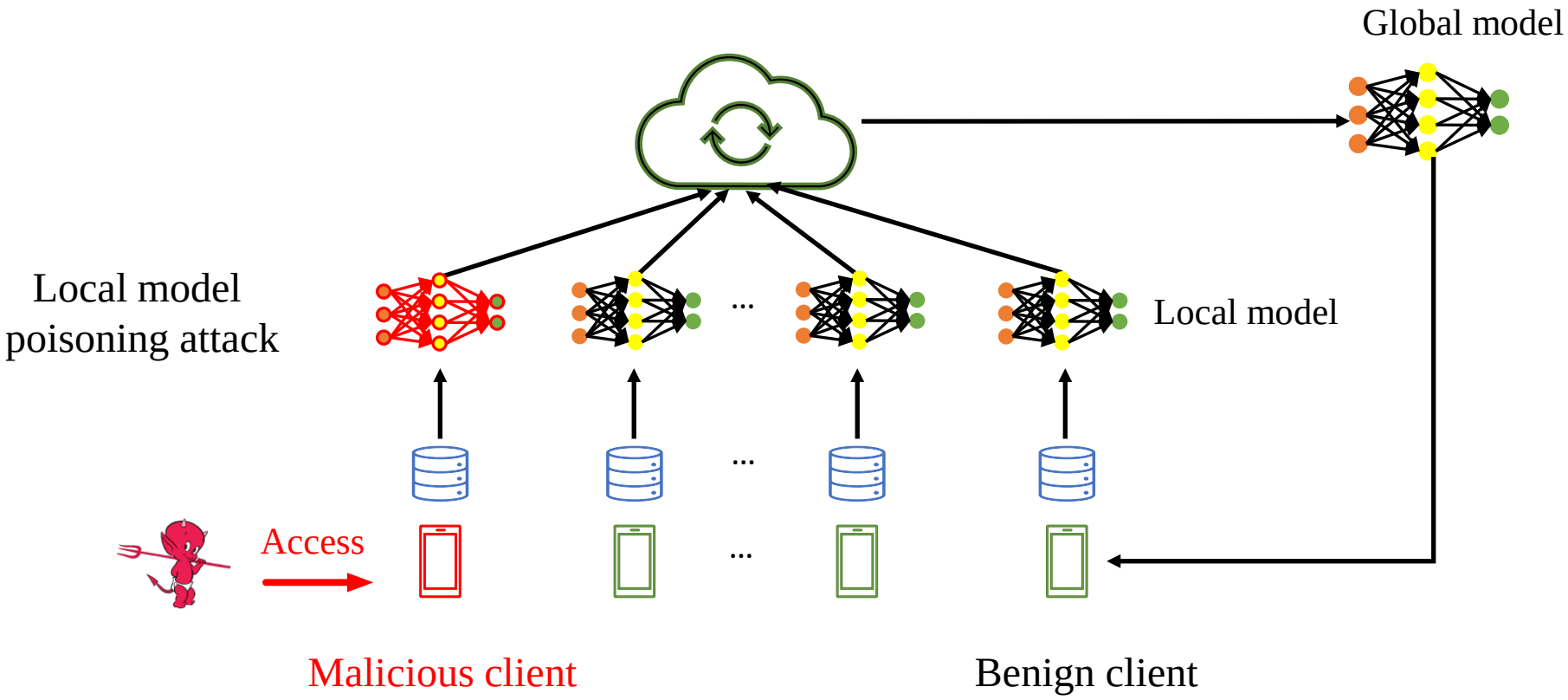


- Fake clients
- Compromised benign clients (by malware infections)

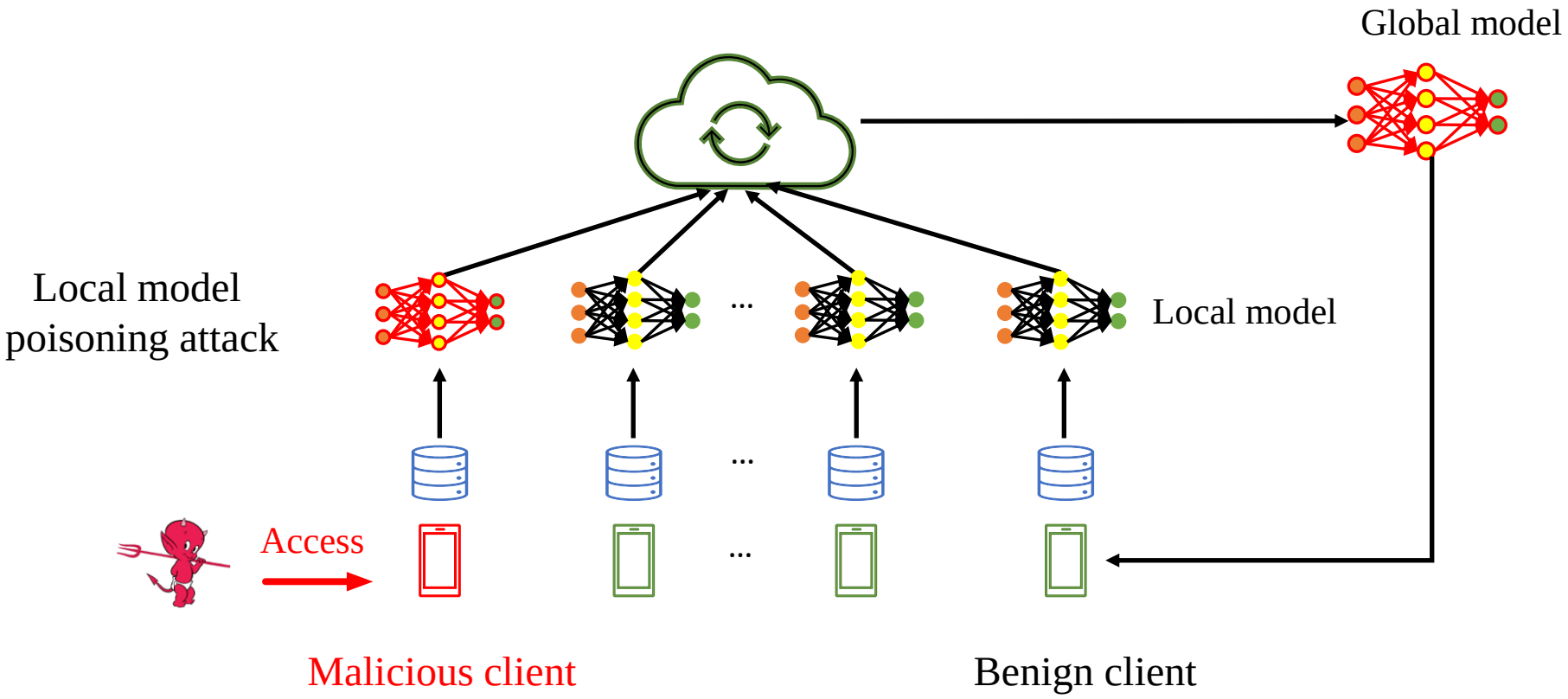
Poisoning Attacks to Federated Learning



Poisoning Attacks to Federated Learning

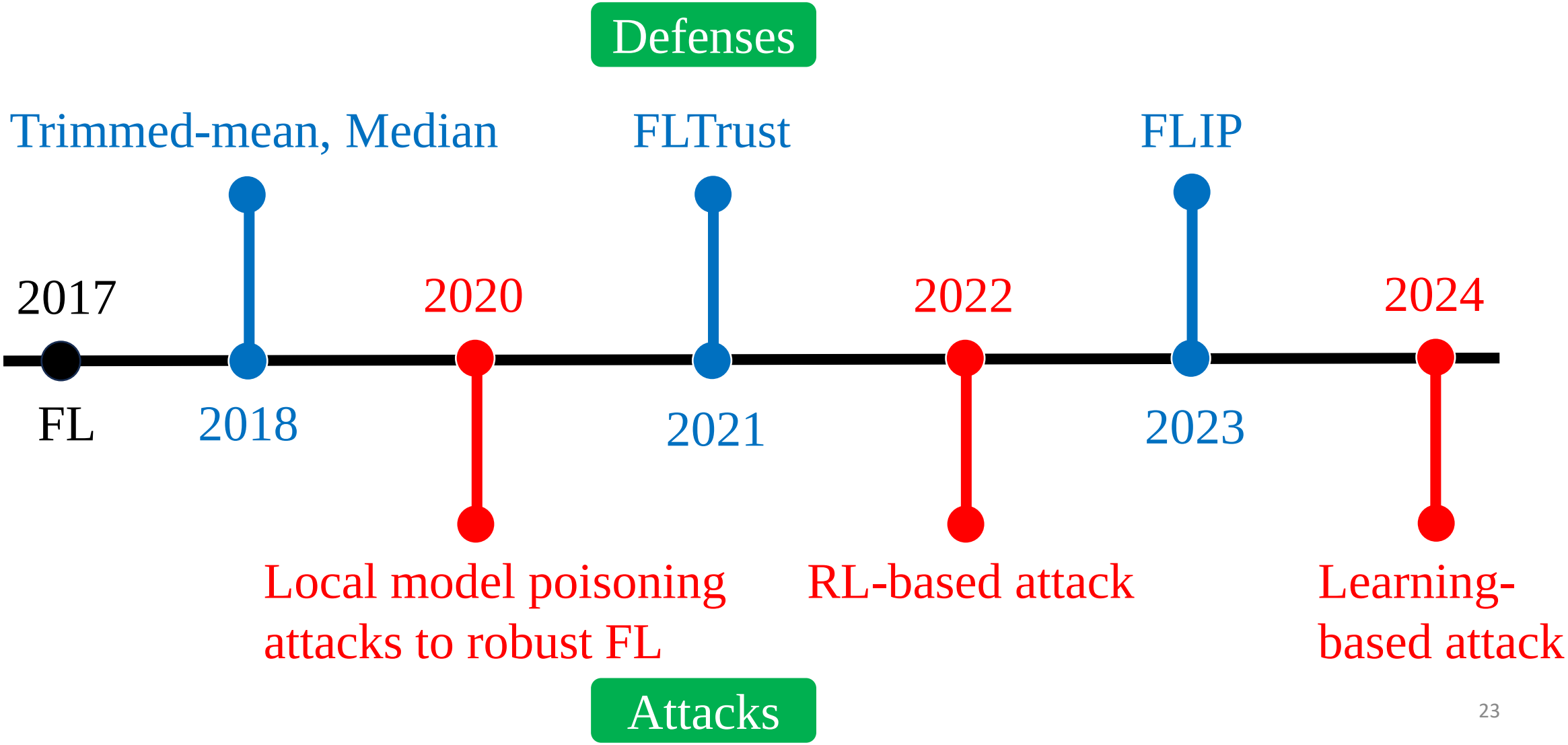


Poisoning Attacks to Federated Learning



Poisoning Attacks to Federated Learning

Cat-and-Mouse Game in Federated Learning Security



Limitations of Existing Defenses

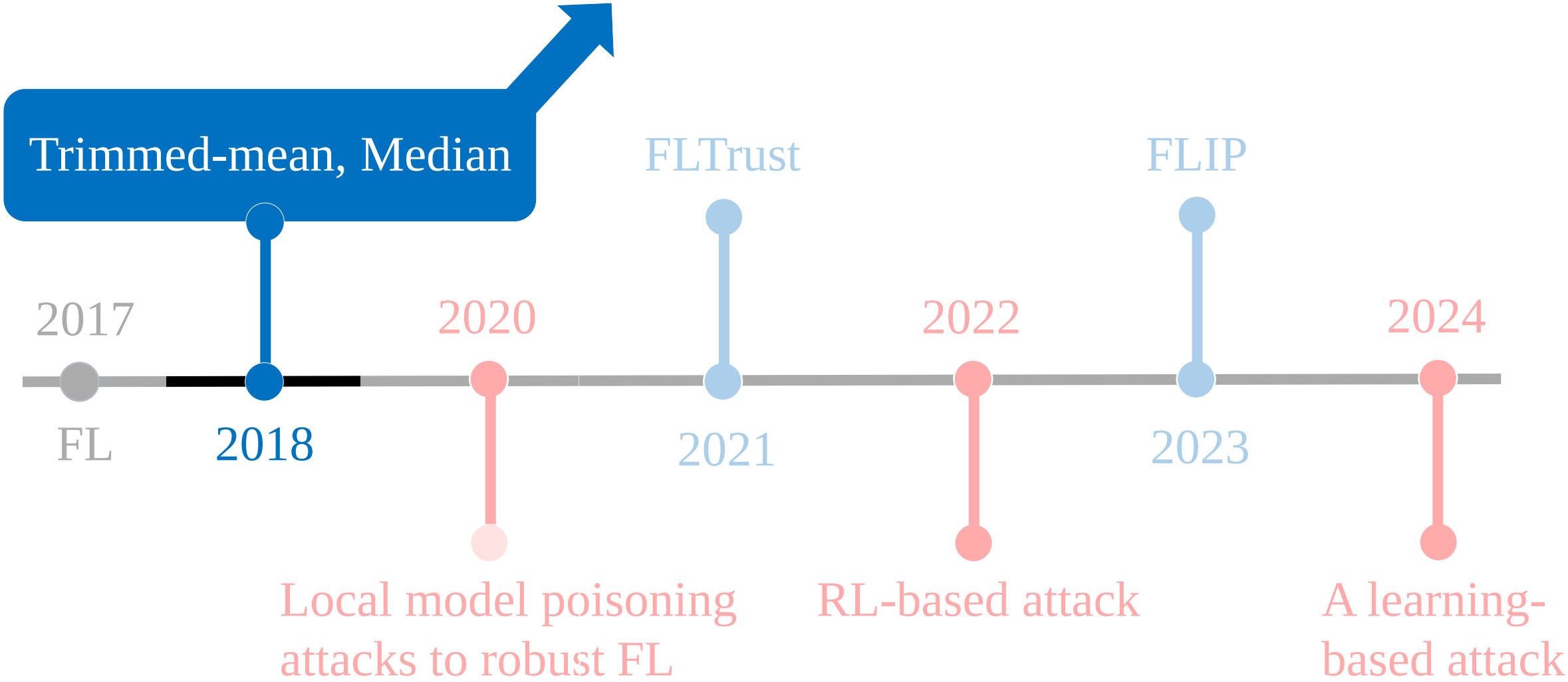
- Unrealistic assumptions
  - Assume the server owns a small trusted dataset
- Increasingly complex
- Remain vulnerable to sophisticated attacks

Do we really need to design new Byzantine-robust aggregation rules?

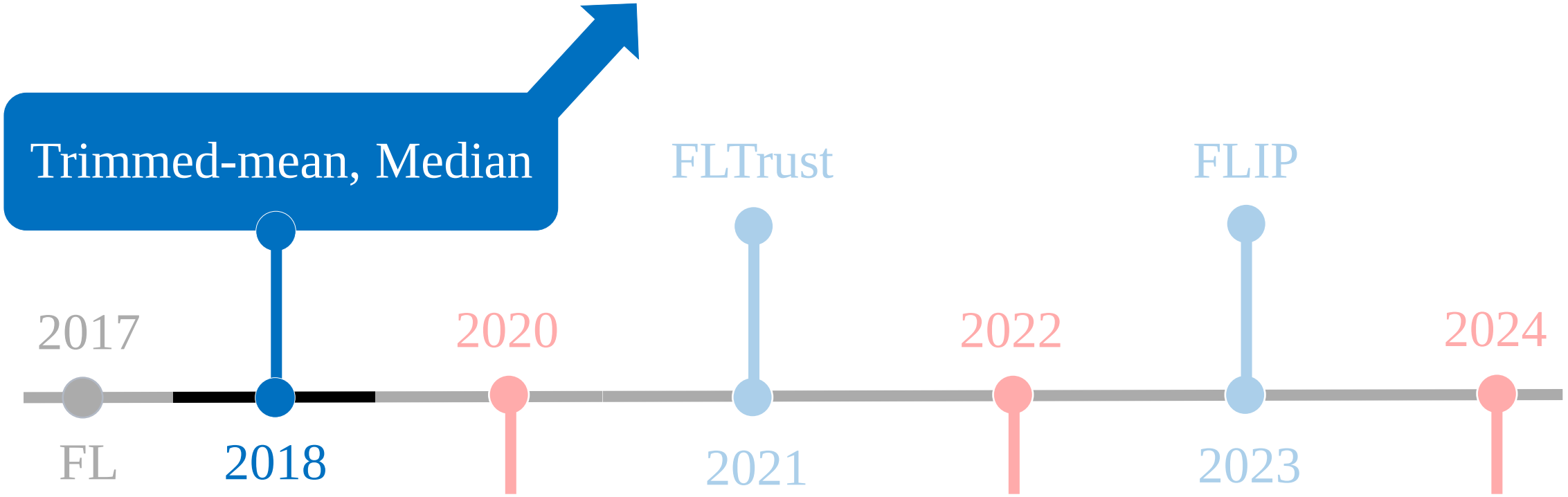
There is no need to design new Byzantine-robust aggregation rules

Federated learning can be secured by enhancing the robustness of existing **foundational aggregation rules**

Foundational Aggregation Rules



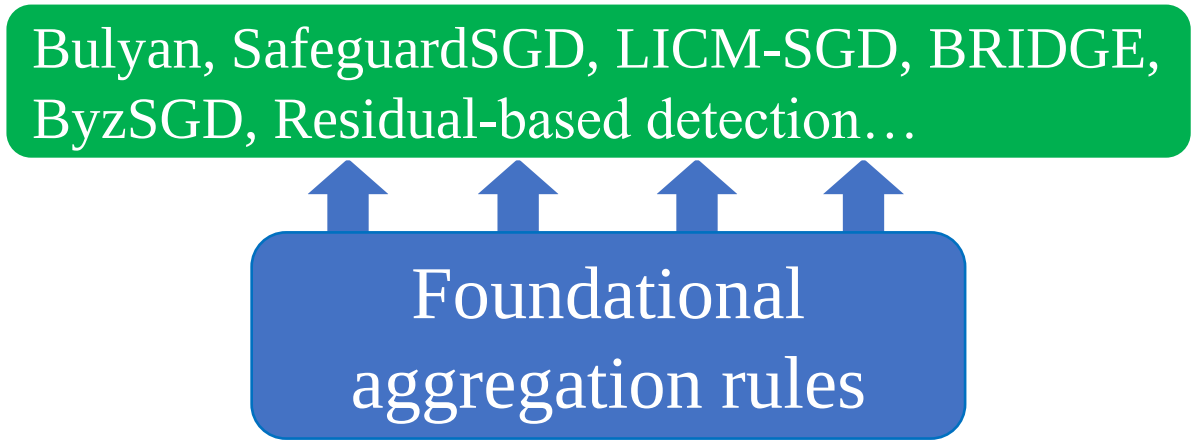
Foundational Aggregation Rules



Foundational aggregation rules mitigate poisoning attacks to some extent, but remain vulnerable to advanced attacks (Fang et al., USENIX Security 2020)

Foundational Aggregation Rules

- Trimmed-mean (ICML 2018)
  - Remove some of the largest and smallest extreme values for each dimension, then average the remaining values
- Median (ICML 2018)
  - Compute the coordinate-wise median of the clients' local model updates
- Simple to implement
- Serve as the backbone for many existing robust aggregation rules



Threat Model

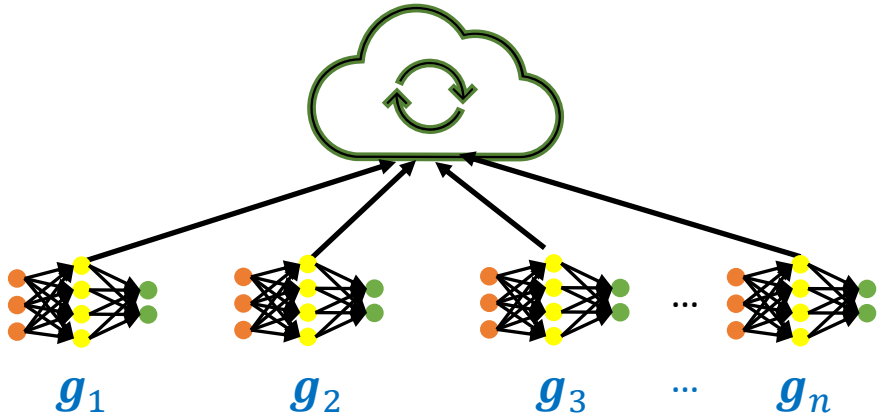
- **Attacker**
  - Attacker’s goal
    - ❖ Untargeted (degrade overall model performance), targeted (manipulate specific outputs)
  - Attacker’s capability
    - ❖ Access to malicious clients, send arbitrary local model updates
  - Attacker’s knowledge (worst-case but realistic attack scenario)
    - ❖ Full knowledge: local model updates on all clients, aggregation rule used by the server
- **Defender**
  - Defender’s goal
    - ❖ Competitive performance, Byzantine robustness, efficiency
  - Defender’s knowledge
    - ❖ Local model updates sent by clients
    - ❖ No knowledge of attack strategy

Our Defense: FoundationFL

- Server generates some **synthetic updates** after receiving local model updates from clients
- Server uses the **foundational aggregation rule**, such as Trimmed-mean or Median, to combine clients' model updates with the synthetic ones

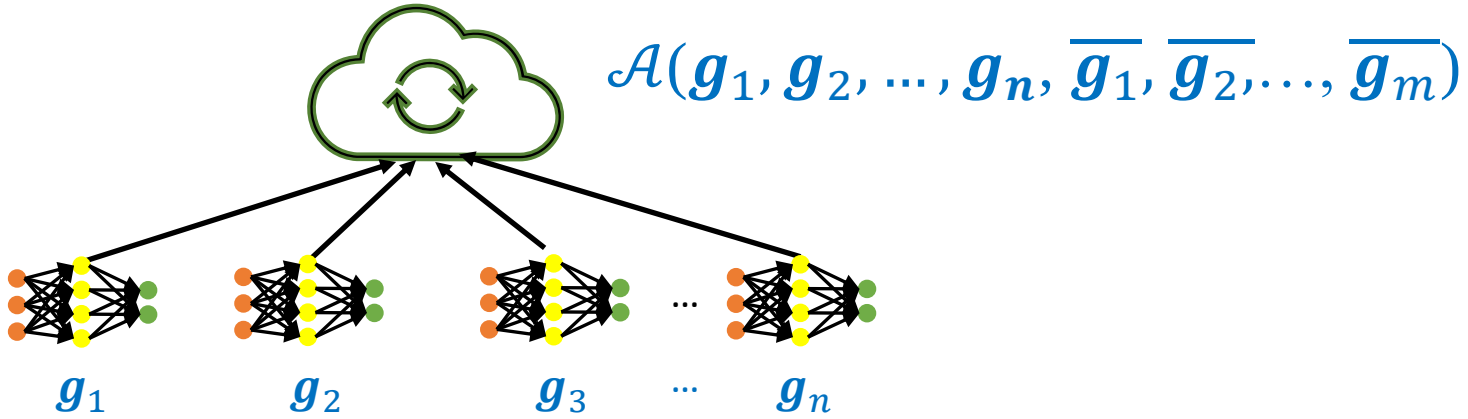
Our Defense: FoundationFL

- Server generates some **synthetic updates** after receiving local model updates from clients
- Server uses the **foundational aggregation rule**, such as Trimmed-mean or Median, to combine clients' model updates with the synthetic ones



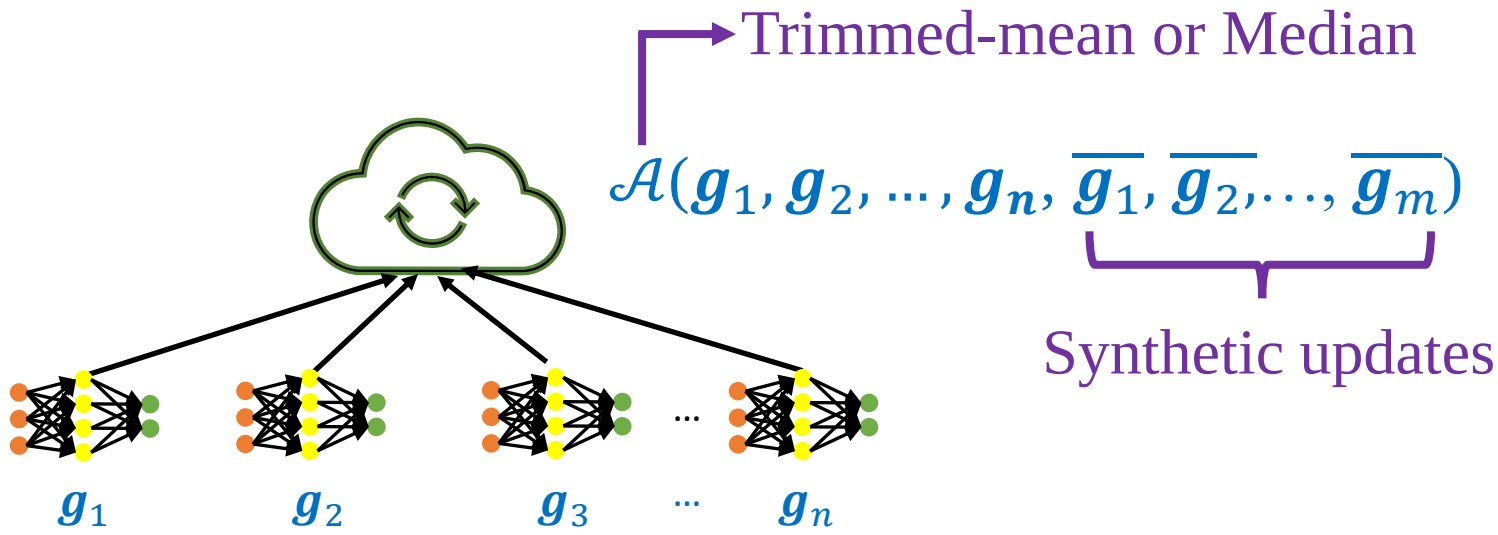
Our Defense: FoundationFL

- Server generates some **synthetic updates** after receiving local model updates from clients
- Server uses the **foundational aggregation rule**, such as Trimmed-mean or Median, to combine clients' model updates with the synthetic ones



Our Defense: FoundationFL

- Server generates some **synthetic updates** after receiving local model updates from clients
- Server uses the **foundational aggregation rule**, such as Trimmed-mean or Median, to combine clients' model updates with the synthetic ones



Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions

Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions

Client 1 $\begin{pmatrix} 0.2 & 0.4 & 0.6 & 0.3 \end{pmatrix}$

Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions

Client 1	0.2	0.4	0.6	0.3
Client 2	0.5	1.5	0.8	0.9

Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions

Client 1	0.2	0.4	0.6	0.3
Client 2	0.5	1.5	0.8	0.9
Client 3	1.6	0.4	1.2	0.3

Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

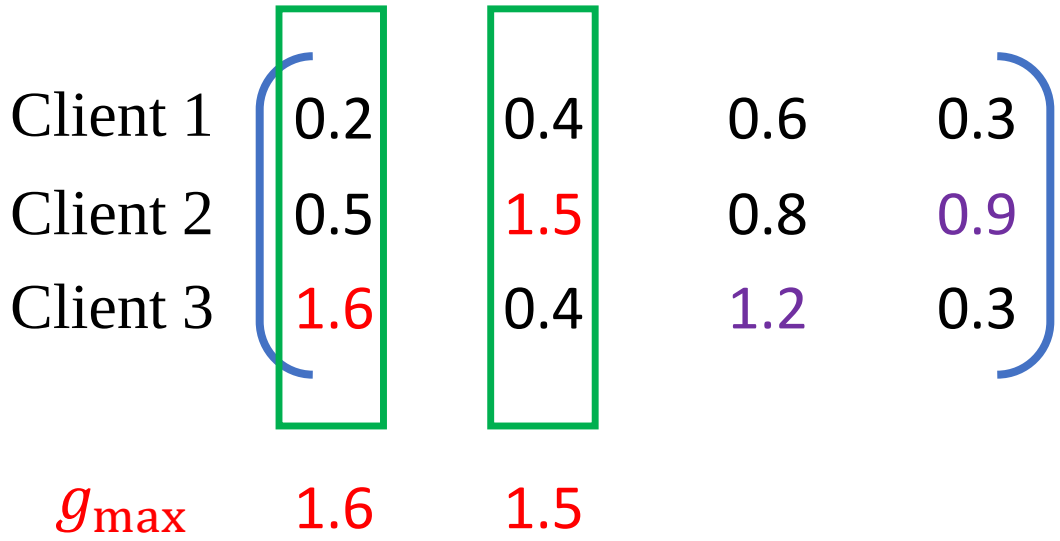
- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions

Client 1	0.2	0.4	0.6	0.3
Client 2	0.5	1.5	0.8	0.9
Client 3	1.6	0.4	1.2	0.3
$g_{\max}$	1.6			

Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

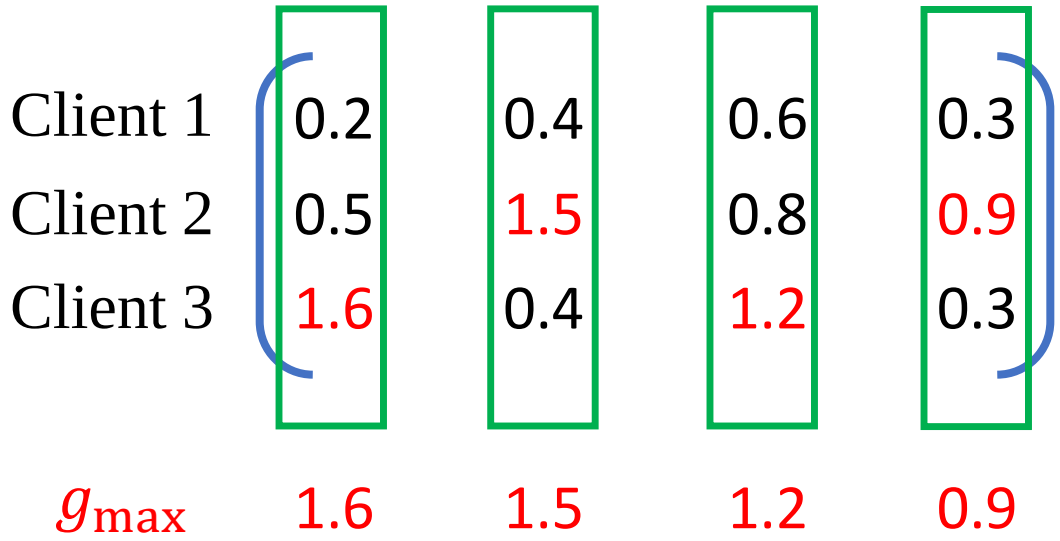
- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions



Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions



Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions
- Select a client's update with the largest distance from $g_{\max}$ and $g_{\min}$ (e.g., distance between g_i and extreme updates can be computed as $\min\{\|g_i - g_{\max}\|, \|g_i - g_{\min}\|\}$)

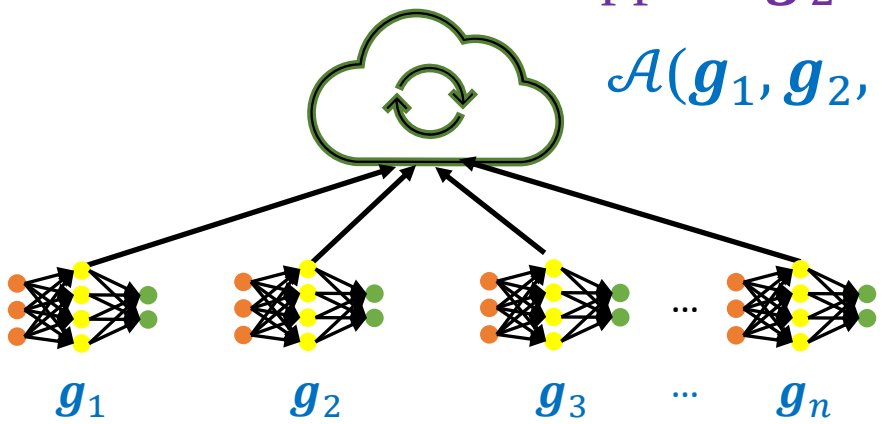
Generating Synthetic Updates

Server identifies a client update that deviates most from extreme updates, and considers it as the synthetic update

- Compute extreme updates: $g_{\max}$, $g_{\min}$ represent the largest and smallest updates across dimensions
- Select a client's update with the largest distance from $g_{\max}$ and $g_{\min}$ (e.g., distance between g_i and extreme updates can be computed as $\min\{\|g_i - g_{\max}\|, \|g_i - g_{\min}\|\}$)

Suppose $\boldsymbol{g}_2$ deviates most from extreme updates

$$\mathcal{A}(\boldsymbol{g}_1, \boldsymbol{g}_2, \cdots, \boldsymbol{g}_n, \underbrace{\boldsymbol{g}_2, \boldsymbol{g}_2 \cdots, \boldsymbol{g}_2}_{\text{Synthetic updates}})$$

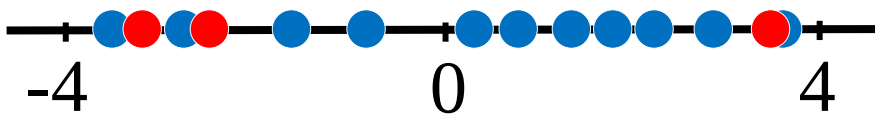


Why FoundationFL Is Effective

- Clients' training data is highly heterogeneous
- Attacker exploits such heterogeneity to launch the attack
- Synthetic updates create a more homogeneous set of updates

Why FoundationFL Is Effective

- Clients' training data is highly heterogeneous
- Attacker exploits such heterogeneity to launch the attack
- Synthetic updates create a more homogeneous set of updates

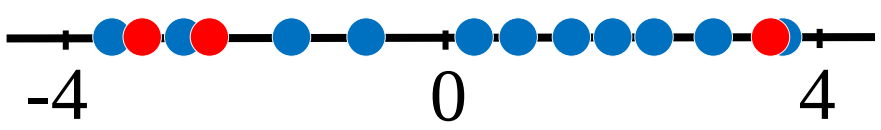


w/o synthetic updates

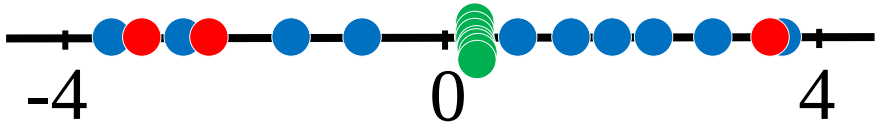
- Benign
- Malicious

Why FoundationFL Is Effective

- Clients' training data is highly heterogeneous
- Attacker exploits such heterogeneity to launch the attack
- Synthetic updates create a more homogeneous set of updates



w/o synthetic updates



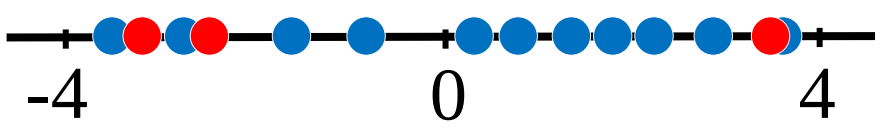
w/ synthetic updates

- Benign
- Malicious
- Synthetic

Why FoundationFL Is Effective

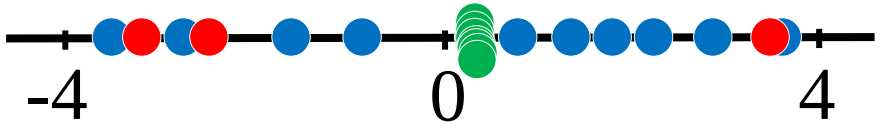
- Clients' training data is highly heterogeneous
- Attacker exploits such heterogeneity to launch the attack
- Synthetic updates create a more homogeneous set of updates

Mean: 0.06
Standard deviation: 2.80



w/o synthetic updates

Mean: 0.14
Standard deviation: 2.29



w/ synthetic updates

Benign
Malicious
Synthetic

Increase the mean and reduce the standard deviation of updates

Experimental Settings

- 100 clients
  - 20 compromised by default
- Datasets:
  - MNIST
  - Fashion-MNIST
  - Human Activity Recognition
  - Purchase
  - Large-scale CelebFaces Attributes
  - CIFAR-10
- Non-IID data distribution
  - Non-IID: not Independently and Identically Distributed
- Server generates 50 synthetic updates in FoundationFL
- 12 poisoning attacks
- 10 comparison aggregation rules

Our FoundationFL is Effective

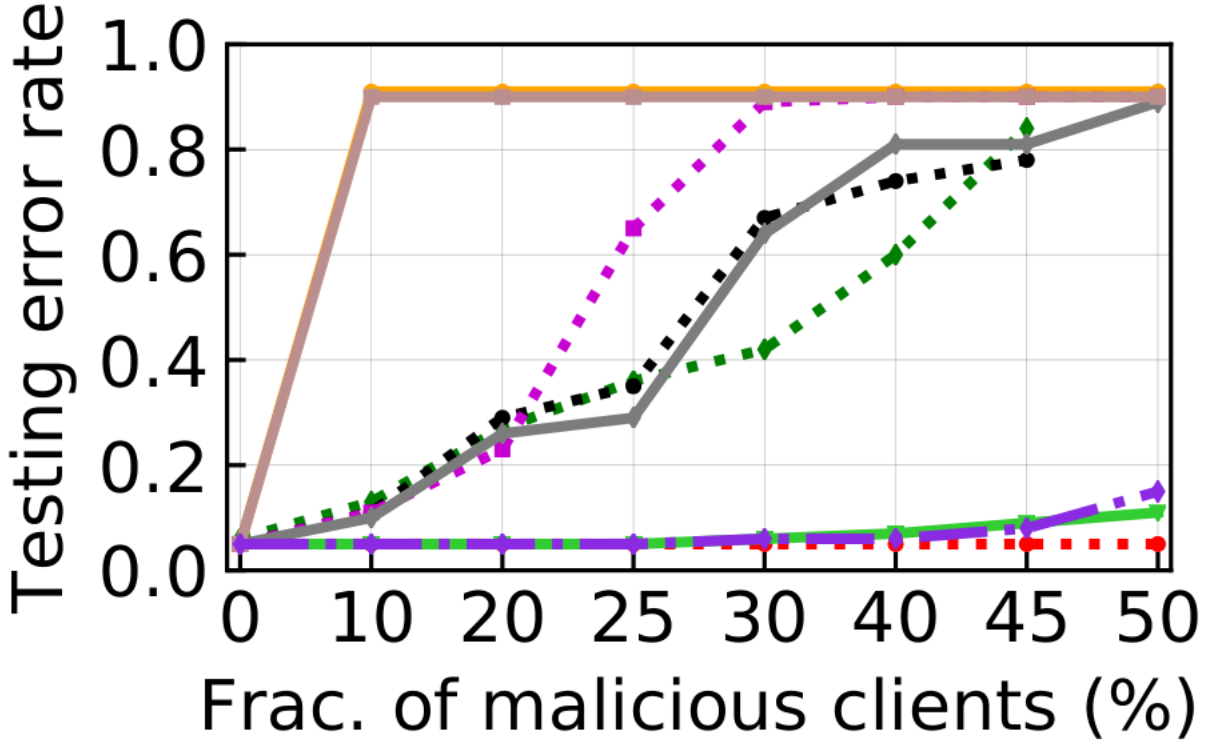
Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.05	0.07	0.90	0.32	0.10	0.90	0.90	0.64 / 0.70
Trim-mean	0.06	0.06	0.06	0.27	0.08	0.19	0.13	0.13 / 0.02
GAS + Trim-mean	0.05	0.05	0.11	0.29	0.07	0.10	0.11	0.43 / 0.47
Gaussian + Trim-mean	0.05	0.11	0.91	0.91	0.05	0.08	0.06	0.91 / 1.00
FoundationFL + Trim-mean	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05 / 0.02
Median	0.05	0.09	0.16	0.23	0.17	0.19	0.23	0.05 / 0.02
GAS + Median	0.05	0.05	0.12	0.26	0.06	0.10	0.10	0.59 / 0.65
Gaussian + Median	0.05	0.90	0.90	0.90	0.05	0.14	0.14	0.91 / 1.00
FoundationFL + Median	0.05	0.05	0.05	0.05	0.07	0.05	0.05	0.06 / 0.02

Our FoundationFL is Effective

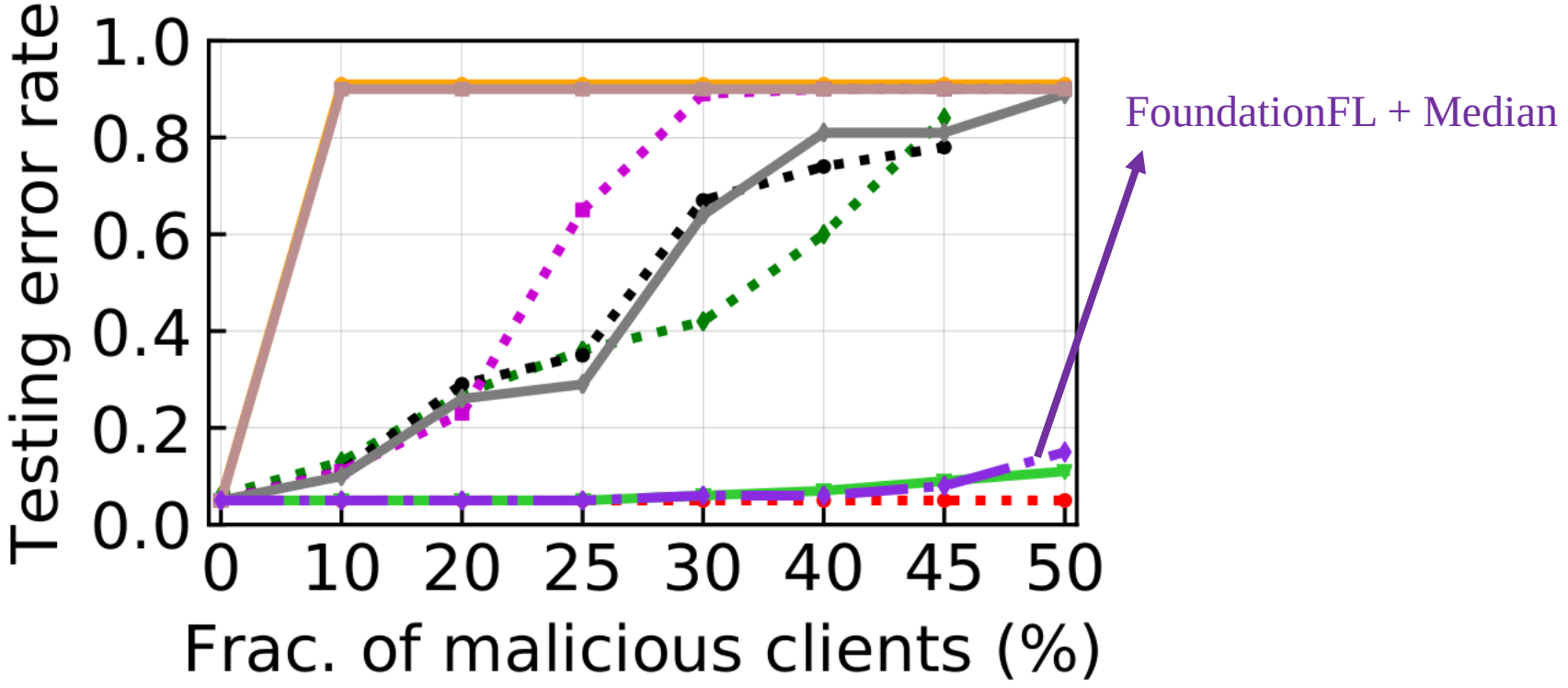
Aggregation rule	No attack	LF attack	Gaussian attack	Trim attack	Krum attack	Min-Max attack	Min-Sum attack	Scaling attack
FedAvg	0.05	0.07	0.90	0.32	0.10	0.90	0.90	0.64 / 0.70
Trim-mean	0.06	0.06	0.06	0.27	0.08	0.19	0.13	0.13 / 0.02
GAS + Trim-mean	0.05	0.05	0.11	0.29	0.07	0.10	0.11	0.43 / 0.47
Gaussian + Trim-mean	0.05	0.11	0.91	0.91	0.05	0.08	0.06	0.91 / 1.00
FoundationFL + Trim-mean	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05 / 0.02
Median	0.05	0.09	0.16	0.23	0.17	0.19	0.23	0.05 / 0.02
GAS + Median	0.05	0.05	0.12	0.26	0.06	0.10	0.10	0.59 / 0.65
Gaussian + Median	0.05	0.90	0.90	0.90	0.05	0.14	0.14	0.91 / 1.00
FoundationFL + Median	0.05	0.05	0.05	0.05	0.07	0.05	0.05	0.06 / 0.02

Use Median rule to combine clients' model updates with the synthetic ones

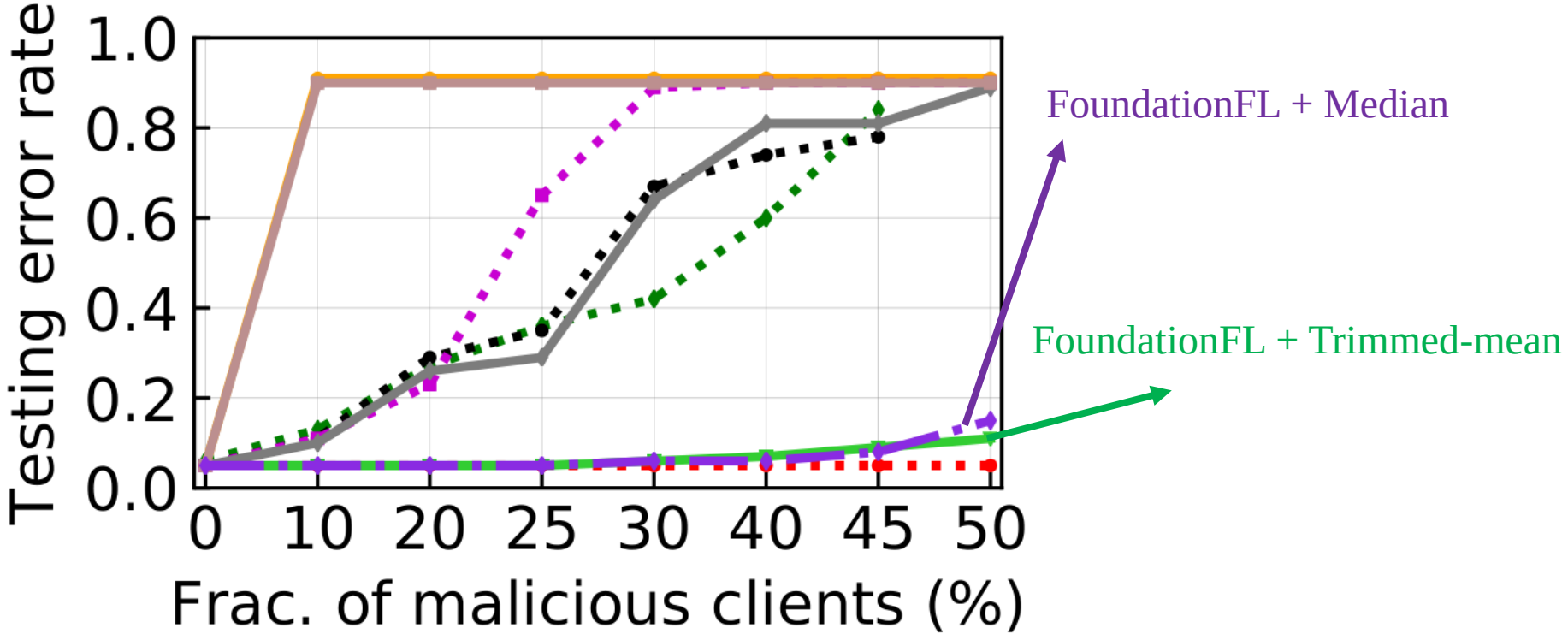
Impact of Fraction of Malicious Clients



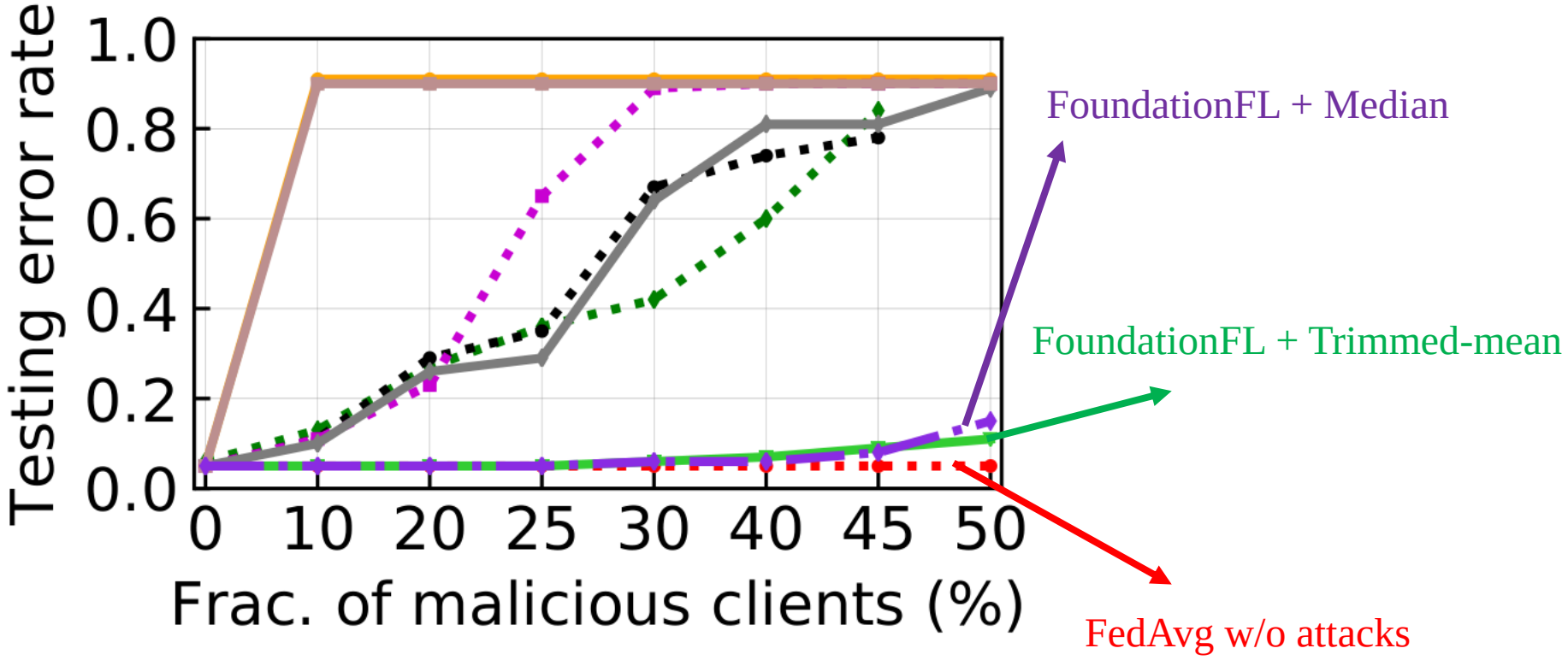
Impact of Fraction of Malicious Clients



Impact of Fraction of Malicious Clients



Impact of Fraction of Malicious Clients



Conclusion

- There is no need to design new Byzantine-robust aggregation rules
- We can secure federated learning by enhancing the robustness of existing foundational aggregation rules

Thank You & Questions?