Federated Learning

Lecture 5 – Attacks & Defenses
Escuela de Ciencias Informáticas (ECI) 2026
Universidad de Buenos Aires

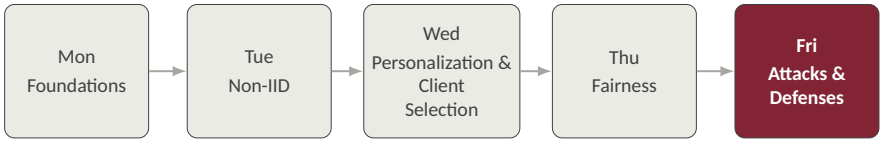
Ioannis Chatzigiannakis
Sapienza University of Rome

ECI 2026 • Buenos Aires • Lecture 5 of 5

SAPIENZA
UNIVERSITÀ DI ROMA

This week schedule

Where today's lecture fits



Running thread. Same FL pipeline all week. Data have been federated, made non-IID, personalized, and fairness-audited – today somebody attacks them, and we defend them.

Today. Who can attack an FL system and how far that gets them; what leaks even when raw data never moves; and which defenses the literature shows actually hold up, at what cost.

Yesterday's assumption, made explicit

Fairness and robustness both assumed *honest* clients

Recall Thursday: aggregation rules were reweighted to be *fair* across honest clients with different data volumes, distributions, or performance levels.

We explicitly assumed every client's update, however skewed, was an honest attempt to improve the global model.

In reality

Some clients are people, devices, or organizations with an incentive to cheat, spy, or disrupt. Once you drop the honesty assumption, the same non-IID-ness that fairness had to accommodate becomes the perfect place for an attacker to hide.

"It's just weights" – is it really safe?

Model updates are not privacy-free by default

FL removes the need to share raw data – but gradients and weights still encode information about the data that produced them.

Gradient inversion.

- An honest-but-curious server can reconstruct training inputs from shared gradients with surprising fidelity, especially for small batches
- Demonstrated on images and text ("*Inverting Gradients*," Geiping et al., NeurIPS 2020)

Membership inference.

- An attacker can decide whether a specific record was part of a client's training set, from model outputs or updates alone (Shokri et al., IEEE S&P 2017)

Take-away

FL changes *what* is exposed (updates instead of raw data), not *whether* anything is exposed. We need a formal guarantee, not just an architectural one.

Formalizing who can do what

Following the notation we introduced on Monday

Recall the FedAvg round: $S_t \subseteq \{1, \dots, K\}$ participate at round t, and

$$w^{t+1} = w^t + \eta \cdot \mathrm{Agg}\big(\{w_i^t\}_{i \in S_t}\big).$$

Split S_t into honest clients H_t and adversarial clients A_t, with $|A_t| \le f$. Adversaries may submit arbitrary $\tilde{w}_i^t$ instead of an honest update.

Two very different goals

Security adversaries corrupt $\mathrm{Agg}(\cdot)$ itself – integrity, availability.
Privacy adversaries never touch the aggregation – they just watch w_i^t, or w^t, more closely than intended.

What does "just weights" actually expose?

Think back to Monday's Gboard example – is it really safe?

Monday's promise: raw data never leaves the client, only model updates do. What could still go wrong once w_i^t is the only thing that travels?

1. **The update is a function of the data.** $w^{t+1} \approx -\eta \nabla_w \ell(w^t; D_i)$ – a gradient is not raw data, but it is not nothing either.

What does "just weights" actually expose?

Think back to Monday's Gboard example – is it really safe?

Monday's promise: raw data never leaves the client, only model updates do. What could still go wrong once w_i^t is the only thing that travels?

1. **The update is a function of the data.** $w^{t+1} \approx -\eta \nabla_w \ell(w^t; D_i)$ – a gradient is not raw data, but it is not nothing either.

What does "just weights" actually expose?

Think back to Monday's Gboard example – is it really safe?

Monday's promise: raw data never leaves the client, only model updates do. What could still go wrong once w_i^t is the only thing that travels?

1. **The update is a function of the data.** $w^{t+1} \approx -\eta \nabla_w \ell(w^t; D_i)$ – a gradient is not raw data, but it is not nothing either.
2. **The server picks the rules.** A curious server can send *different* w^t to different clients and compare their responses (model inconsistency).

What does “just weights” actually expose?

Think back to Monday’s Gboard example – is it really safe?

Monday’s promise: raw data never leaves the client, only model updates do. What could still go wrong once w_i^t is the only thing that travels?

1. **The update is a function of the data.** $w^{t+1} \approx -\eta \nabla_w \ell(w^t; D_i)$ – a gradient is not raw data, but it is not nothing either.
2. **The server picks the rules.** A curious server can send *different* w^t to different clients and compare their responses (model inconsistency).
3. **Multiple rounds accumulate.** A single gradient may reveal little; watching the same client for hundreds of rounds is a very different threat.

What does “just weights” actually expose?

Think back to Monday’s Gboard example – is it really safe?

Monday’s promise: raw data never leaves the client, only model updates do. What could still go wrong once w_i^t is the only thing that travels?

1. **The update is a function of the data.** $w^{t+1} \approx -\eta \nabla_w \ell(w^t; D_i)$ – a gradient is not raw data, but it is not nothing either.
2. **The server picks the rules.** A curious server can send *different* w^t to different clients and compare their responses (model inconsistency).
3. **Multiple rounds accumulate.** A single gradient may reveal little; watching the same client for hundreds of rounds is a very different threat.
4. **Any client can be the attacker.** An insider client can craft its own updates to bias, backdoor, or destabilize the model others rely on.

One root cause

FL removes the need to centralize data – it does not remove the need for integrity checks or confidentiality guarantees. Those have to be engineered in separately.

5/29 Ioannis Chatzigiannakis | Federated Learning | ECI 2026 | Lecture 5

Poisoning: the easiest attack to imagine

Same aggregation rule, malicious inputs

Data poisoning. A malicious client alters its own local data before training on it.

Concrete instances.

- *Untargeted*: random label flips or noisy pixels – degrades overall accuracy, no specific target.
- *Targeted*: stop signs mislabeled as speed-limit signs in an autonomous-driving dataset.
- *Clean-label*: only features are perturbed, labels stay correct – much harder to catch than *dirty-label* poisoning, where labels are swapped outright.

Why start here

It is the attack that needs the least access: no control over aggregation, no cryptographic tricks – just a client willing to lie about its own data.

Model poisoning: a stealthier cousin

Fang et al., USENIX Security 2020; Baruch et al., NeurIPS 2019

Instead of poisoning data, the adversary crafts the update $\tilde{w}_i^t$ directly.

Strategy	Idea	Result
Non-directional	maximize deviation from honest mean	large accuracy drop, easy to flag
"A Little Is Enough"	exploit honest-update *variance*	bypasses robust aggregation

Model poisoning routinely outperforms data poisoning against Byzantine-robust defenses, because it needs only a handful of colluding clients and limited knowledge of honest data.

Why non-IID makes this worse

Honest updates from a skewed client already look like outliers – the attacker hides inside variance that was there anyway.

When the goal is disruption, not deception

Jamming, evasion, straggling, dropout

Attack	Phase	Effect
Jamming	training	wireless interference blocks update delivery
Evasion	inference	adversarial input fools the *deployed* model
Straggling	training	compromised clients slow every round down
Dropout	training	clients vanish at the worst possible moment

Straggling and dropout are not always malicious – but an adversary can trigger either one deliberately, and a defense that only looks for *wrong* updates will miss both.

Backdoors, Sybils, and free-riders

Targeted, colluding, and exploitative adversaries

- **Backdoor.** The global model behaves normally on clean inputs, misclassifies whenever a trigger τ is present. Fung et al. showed 2 Sybil clients made 96.2% of MNIST "1"s read as "7"; Chameleon and A3FL make triggers *durable* even after the attacker leaves.
- **Sybil.** One adversary runs many fake client identities to inflate its aggregation weight.
- **Free-riding.** A client returns noise or stale weights, contributes nothing, but still benefits from every global update (*anonymous* vs. *selfish* free-riders).

Common thread

All three exploit the same trust assumption FedAvg makes: that every submitted update came from one honest client with a genuine local dataset.

Naive mistake: trusting the mean

What happens if the server just runs plain FedAvg

Suppose the server aggregates as usual, unaware that f of the $|S_t|$ clients are adversarial:

$$w^{t+1} \leftarrow w^t + \eta \sum_{k \in S_t} \frac{n_k}{n} w_k^t$$

A *single* unbounded $\tilde{w}_i^t$ can move w^{t+1} arbitrarily far – the arithmetic mean has no built-in resistance to even one adversarial outlier.

The same n_k-weighting that fixed quantity skew on Tuesday now works against you: a large adversarial client's poisoned update gets *more* trust, not less.

The fix, in one sentence

Replace the mean with an estimator that is provably insensitive to up to f arbitrary inputs – a *robust aggregation operator*.

Robust aggregation operators

Replacing the mean with something outlier-resistant

- **Trimmed Mean.** Drop the top/bottom $\beta\%$ of each coordinate, average the rest.
- **Median / GeoMed.** The geometric median tolerates up to 50% malicious clients; AutoGM adds automatic re-weighting for skewness.
- **Krum / Multi-Krum.** Pick the update(s) closest to their $n - f$ nearest neighbours – assumes bounded skewness among honest clients.
- **Bulyan.** Compress with Krum or GeoMed first, *then* aggregate robustly – more expensive ($O(dn^2)$), more resistant.

Non-IID caveat

Every method above assumes "different" means "suspicious". Under real non-IID data, an honest-but-skewed client looks exactly like an attacker to Krum or Trimmed Mean.

Does this actually work? The empirical evidence

Jimenez-Gutierrez et al., Information Fusion 2025 – §3.3

Defense	Global accuracy under attack	Cost
Bulyan	up to 85% (20% adversaries)	$O(dn^2)$, slow at scale
Trimmed Mean	78%	cheap, simple
Krum	75%	fails on RNNs, IPM/ALIE attacks
FLTrust	$>$90% clean, $<$5% backdoor success	needs a small trusted set
FLAME	$\sim$88%, non-IID robust	adaptive noise injection

Takeaway

No single operator dominates: Bulyan wins on raw robustness, FLTrust wins when you can afford a trusted validation set, FLAME wins under heavy non-IID skew.

Beyond robust aggregation

Four more defense families, each for a different failure mode

- **Asynchronous / coded schemes** (FedAAM, CodedPaddedFL): do not wait for stragglers or jammed clients – 95% MNIST accuracy, 18$\times$ speed-up, at roughly 2$\times$ bandwidth cost.
- **Pruning**: remove neurons silent on clean data – cuts a FashionMNIST backdoor success rate from 99.7% to 2%, though pruning-aware attackers can adapt.
- **Adversarial training**: clients train on local adversarial examples, formulated as a min-max problem – pFedDef gains $\sim$60% robustness to PGD attacks on CIFAR.
- **Personalized solutions**: FoolsGold (cosine-similarity Sybil detection), LeadFL (Hessian regularization), PASS (free-rider auditing), Sageflow (staleness-aware weighting).

No defense here stops jamming – that needs the asynchronous/coded family instead.

Stepping back: two different kinds of harm

Jimenez-Gutierrez et al., 2025 survey – unified taxonomy

Security attacks corrupt *what the model becomes*. Privacy attacks corrupt *what the model reveals* – without ever touching aggregation.

- **Gradient manipulation.** Inversion, suppression, canary gradients.
- **Inference.** Membership inference, reconstruction through inference, GAN-based inference, model inconsistency.
- **Interception.** Eavesdropping, unintentional data leakage.

Why the distinction matters

A perfectly Byzantine-robust aggregator can still leak every client's raw data through the gradients it faithfully aggregates. Robustness and confidentiality are separate axes, and defending one does not defend the other.

Gradient inversion: reconstructing data from "just weights"

Geiping et al., NeurIPS 2020; Kariyappa et al., ICML 2023

An attacker observing a gradient-like signal g^t solves

$$\min_{x,y} \left\| \nabla_w \ell(w^t; x, y) - g^t \right\|_2^2$$

to recover the training example (x, y) that produced it.

Concrete instances.

- Cocktail Party Attack: independent component analysis recovers private inputs even from large-batch aggregated gradients.
- E2EGI: iteratively inverts gradients across multiple communication rounds.

Mitigating factors

Larger batches, more local epochs, and gradient clipping all reduce reconstruction fidelity – but do not eliminate the risk, especially against a determined, multi-round attacker.

Two workhorse privacy defenses

Recall Monday's closing slide – now with the mechanics filled in

Differential Privacy

$\mathcal{M}$ is (ε, δ)-DP if for neighbouring datasets D, D' and any event S: $\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta$. Instantiated via per-example gradient clipping plus Gaussian noise (DP-SGD).

Secure Aggregation / MPC

The server only ever observes $\sum_{i \in S_t} w_i^t$, never an individual w_i^t. Exact cryptographic aggregation – **zero** accuracy cost, but real computational and communication overhead.

Neither restores integrity: a Byzantine client can still hide a poisoned update inside an encrypted or noised aggregate.

Membership inference and canary gradients

Shokri et al., IEEE S&P 2017; Maddock et al. (CANIFE), 2022

Membership inference. Given access to the trained model, decide whether a specific record z was part of some client's training set – a binary hypothesis test on confidence scores or loss values.

Canary gradient. Inject a crafted perturbation into a client's update, then watch whether it reappears in the aggregate – CANIFE showed the empirical per-round privacy loss is often *tighter* than the theoretical DP bound predicts.

And three more worth knowing by name

GAN-based inference: a malicious client trains a generator to mimic another client's data distribution. **Model inconsistency**: a curious server sends different model versions to different clients to compare responses. **Eavesdropping**: passive or active interception of unencrypted traffic between clients and server.

Differential privacy: a formal guarantee

Bounding what an observer can learn about any single client

A randomized mechanism $\mathcal{M}$ is (ε, δ)-differentially private if for any two neighboring datasets D, D' differing in one record, and any measurable output set S:

$$\Pr[\mathcal{M}(D) \in S] \;\leq\; e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta$$

Reading it. The output distribution barely changes whether or not any one individual's record is included – an observer cannot confidently tell.

The privacy budget ε.

- Smaller $\varepsilon \Rightarrow$ stronger indistinguishability $\Rightarrow$ more noise $\Rightarrow$ worse utility
- δ is the (small) probability the guarantee fails completely

Foundational reference: Dwork & Roth, *The Algorithmic Foundations of Differential Privacy*, 2014.

Making FL differentially private: DP-SGD / DP-FedAvg

Clip, then add calibrated noise

Two mechanisms calibrate noise to a query's *sensitivity* Δ:

- **Laplace mechanism**: noise $\sim \text{Lap}(\Delta/\varepsilon)$, gives pure ε-DP
- **Gaussian mechanism**: noise $\sim \mathcal{N}(0, \sigma^2)$ with $\sigma \propto \Delta\sqrt{2\ln(1.25/\delta)}/\varepsilon$, gives (ε, δ)-DP

DP-SGD (Abadi et al., ACM CCS 2016):

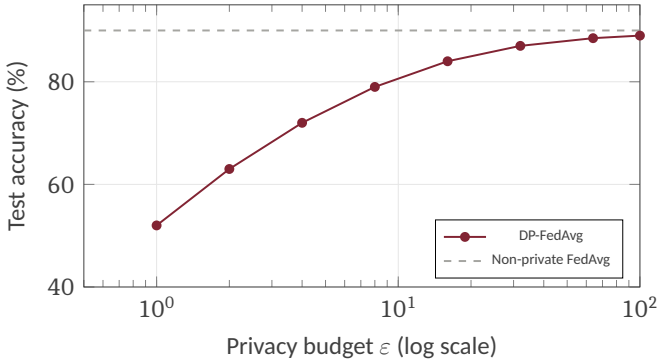
1. Compute per-example gradients
2. **Clip** each to bound its ℓ_2 norm by C (bounds sensitivity)
3. **Add Gaussian noise** $\mathcal{N}(0, \sigma^2 C^2 I)$ to the summed/averaged gradient

Local vs. central DP in FL.

- *Local DP*: each client noises its own update before upload – no trust in the server
- *Central DP*: the server clips/noises after aggregation – less noise, but requires trusting the aggregator (or combining with secure aggregation)

The privacy–utility trade-off

Accuracy as a function of the privacy budget ε



Small ε (strong privacy) costs several points of accuracy; the curve flattens once ε is large enough that noise becomes negligible relative to the gradient signal. Illustrative shape, consistent with reported DP-FedAvg behavior in the FL privacy literature.

The rest of the crypto toolbox

Beyond DP: what cryptography adds

- **Homomorphic Encryption (HE).** Compute directly on ciphertexts – strong confidentiality, 99.95% accuracy reported on MNIST, but heavy compute cost.
- **Secret Sharing.** Split a key across t clients – collusion-resistant up to $t-1$ colluders.
- **MPC.** Jointly compute a function without revealing inputs – FLAG scales to thousands of clients.
- **Zero-Knowledge Proofs.** Prove a computation was done correctly without revealing the data – powerful, but currently costly at FL scale.

Quantitative snapshot

DP-MPC hybrids: 16–182× faster than naïve MPC while keeping DP guarantees, and 56–794× more communication-efficient in some configurations.

Which defense stops which attack?

No silver bullet – combine complementary families

Defense	Poisoning	Backdoor	Jamming
Robust aggregation	✓	✓	
Asynchronous schemes			✓
Pruning		✓	
Adversarial training	✓		
Personalized solutions	✓	✓	

Recommended pairings. Bulyan + FedAAM for integrity *and* liveness; Trimmed Mean + pFedDef for poisoning *and* evasion; FLAME when the deployment is heavily non-IID.

Which framework should you actually use?

Jimenez-Gutierrez et al., 2025 – 13-framework comparison

- **FEDML**: most complete – DP, HE, and built-in attack *and* defense simulation (`FedMLAttacker` / `FedMLDefender`).
- **Flower** (this week's lab tool): scales to millions of simulated clients, strong secure-aggregation support, but no built-in attack simulator.
- **FATE / PySyft / TFF**: mature HE and MPC support, good for cross-silo and vertical FL.

A gap you could fill

No surveyed framework implements GeoMed-style defenses natively, and support for vertical/split FL remains incomplete almost everywhere.

Where this matters in practice

Domains driving secure/private FL research

- **Text prediction** (Gboard-style): poisoning, GAN-inference, gradient inversion on keystroke models.
- **Healthcare**: multi-hospital imaging and EHR data, membership inference on patient records – HE and MPC are the primary defenses.
- **Finance**: fraud detection, GAN-based poisoning of shared risk models.
- **Intrusion detection systems**: a fastest-growing application area, dominated by label-flipping attacks on IoT logs.

The pattern from Monday, again

Every one of these domains uses FL *because* its data is sensitive, regulated, or too large to centralize – and that same sensitivity is exactly what every attack in this lecture targets.

Open problems and where the field is going

Security, privacy, and their intersection

- **Joint security–privacy modeling.** A poisoning attack can increase sensitivity to gradient inversion; heavy DP noise can hide adversarial behaviour from anomaly detectors. Most defenses are still designed and evaluated in isolation.
- **Quantitative privacy–utility trade-offs.** No unified mathematical framework yet links privacy budget, computational overhead, and accuracy.
- **Federated foundation models.** Federated fine-tuning of LLMs/VLMs introduces new leakage channels (prompts, cross-modal reconstruction) and new attack surfaces.
- **Standardized, reproducible benchmarks.** Most domains still lack realistic, compound-attack evaluation pipelines.

Table of Contents

This afternoon's lab

Attack and defend your own pipeline

Environment. The same Flower + PyTorch codebase from Monday through Thursday.

Steps.

1. Implement an untargeted label-flipping poisoning attack on 20% of clients.
2. Measure the global accuracy drop under plain FedAvg.
3. Swap in Trimmed Mean and Krum aggregation; re-measure accuracy and convergence.
4. Add a stealthy backdoor trigger to one client; check its success rate before and after norm-clipping or pruning.

Deliverable

A plot comparing FedAvg vs. Trimmed Mean vs. Krum accuracy under 0%, 10%, 20%, 30% malicious clients, plus one paragraph on which defense you would deploy for your own non-IID partition from Tuesday's lab, and why.

Course recap: five lectures, one thread

From theory to practice to advanced topics

1. **Foundations.** FL trades data movement for model movement – that shifts the deployment cost, it does not automatically guarantee privacy or security.
2. **Non-IID.** Heterogeneity is quantifiable, and quantity skew is the one flavor that correct weighting almost fully absorbs.
3. **Personalization & Client Selection.** One global model is often the wrong target; who you select shapes both performance and attack surface.
4. **Fairness.** Aggregation choices that help robustness can hurt fairness, and fairness attacks are themselves a poisoning vector.
5. **Attacks & Defenses.** No defense is free and none is complete – robust aggregation, DP, secure aggregation/MPC, and personalized solutions must be combined and matched to your actual threat model.

Table of Contents

Thank you

Hope you enjoyed !

ichatz@diag.uniroma1.it