JuLenny Data Privacy

Federated Average (FedAvg)

How private cross-organization model training works

The scenario

Two telecom carriers (Carrier A and Carrier B) run the same AI model to detect network anomalies: dropped calls, unusual traffic spikes, suspicious routing patterns. Each carrier has trained the model on their own network data.

Neither carrier will share their raw network data with the other. They will also not share their trained model weights, because those weights reveal patterns about their network infrastructure, capacity, and weak points.

The problem

Each carrier trains its model on its own network. A model trained on BOTH networks would be much better at catching anomalies (e.g. fraudulent routing patterns that span both networks, coordinated attacks, shared infrastructure issues). But sharing the model weights is almost as sensitive as sharing the raw data.

How FedAvg solves this

Step 1: Each carrier trains locally

Carrier A trains the anomaly detection model on its own network data, producing a list of numbers (weights) that represent what the model learned. For example:

```
Carrier A weights: [0.42, -1.30, 0.87, 0.03, -0.55, ...]
```

Carrier B does the same on its own data:

```
Carrier B weights: [0.15, -0.90, 1.12, -0.21, 0.33, ...]
```

These numbers ARE the model. Anyone who sees them can reconstruct what the model learned and infer things about the training data.

Step 2: Each carrier encrypts their weights

Using the JuLenny toolkit on their own machine, each carrier encrypts their weight list. The encrypted weights are uploaded to the JuLenny platform. Nobody, not even JuLenny, can see the raw numbers.

Step 3: Compute the scaling factors

The average should weight each carrier's contribution by how much data they trained on. If Carrier A trained on 10,000 data points and Carrier B on 5,000:

```
Scale A = 10,000 / 15,000 = 0.667
Scale B =  5,000 / 15,000 = 0.333
```

These scaling factors are plain numbers (not secret), uploaded alongside the encrypted weights.

Step 4: The platform computes the weighted average

JuLenny multiplies each carrier's encrypted weights by their scaling factor, then adds the results:

```
Global weights = 0.667 * [Carrier A weights] + 0.333 * [Carrier B weights]
```

All this math happens entirely on encrypted data. The platform never sees the actual weight values.

Step 5: Decrypt the global model

The averaged global weights are released (after both carriers approve the decryption). Each carrier downloads and decrypts them, then loads the global weights into their local model. Now both carriers have a model that learned from both networks.

Step 6: Repeat

They run this every day, week, or hour. In each round:

1. Both carriers train locally on new data
2. Encrypt updated weights
3. Upload to the platform
4. Platform computes the average
5. Both download the improved global model
6. The model keeps getting better across both networks

JuLenny Data Privacy

What makes this special

No data sharing. Neither carrier sees the other's network data, model weights, or training patterns.

No trust required. Even if JuLenny's cloud is fully compromised, the encrypted weights cannot be decrypted. The joint key is split between the two carriers; neither side (and not JuLenny) holds the full key.

Linear-only circuit. FedAvg only uses multiplication by a plain number and addition. No complex operations, no rotations, no bootstrapping. This is the simplest possible FHE computation, which means it runs fast and the encrypted result has virtually no noise.

Iterative. The more rounds they run, the better the model gets. Each round is an independent computation on the platform.

Beyond telecom

The same function works for any scenario where two organizations want to combine ML models without sharing data:

- **Cross-bank fraud detection**. Two banks each train a fraud model on their own transaction data. The combined model catches patterns that span both banks (e.g., a fraudster moving money between accounts at different institutions).
- **Multi-site clinical trials**. Two hospitals each train a diagnostic model on their patient data. The combined model is more accurate because it learned from a larger, more diverse patient population. No patient records leave either hospital.
- **Defense and cyber**. Two agencies each train a threat detection model on their own intelligence. The combined model catches coordinated attacks that neither agency could see alone.
- **Supply chain**. Two manufacturers each train a quality prediction model on their production data. The combined model improves defect detection across both production lines.

Technical details

- **Scheme:** CKKS (approximate real-number arithmetic, suitable for floating-point model weights)
- **Multiplicative depth**: 1 (one plaintext-times-ciphertext multiplication per input)
- **Rotation keys**: Not needed
- **Relinearization keys**: Not needed
- **Max vector size**: 8,192 weights per ciphertext (larger models require chunking across multiple ciphertexts)
- **Precision**: ~30 bits (more than sufficient for ML weight averaging)