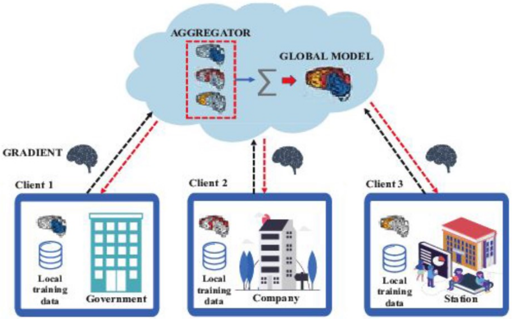
Leveraging Analog Codes for Privacy and Robustness in Federated Learning

Harshan Jagadeesh

Department of Electrical Engineering
Indian Institute of Technology Delhi, India

Joint work with Usayd Shahul
(MS(R) student, IIT Delhi)

Overview

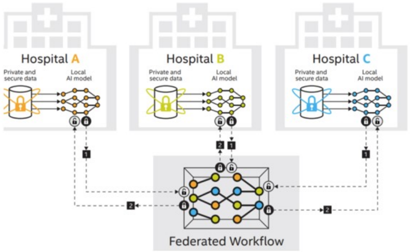


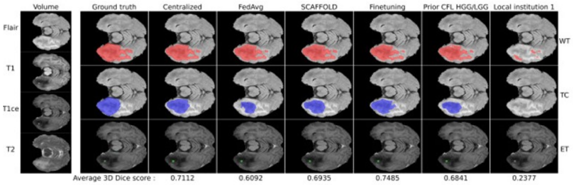
Lack of Data **Privacy** **Accurate Models** **Robust models**

Overview

Each hospital or medical institution acts as a silo. Example Projects:

- *Federated Tumor Segmentation (FeTS)* initiative: Trains brain tumor segmentation models across hospitals.
- *NVIDIA Clara*: Used in cross-institutional FL for medical imaging.



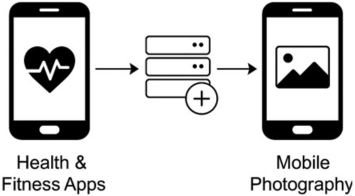


Other Usecases:

- Banking and financial fraud detection
- Radio resource management (RRM) in shared smart grid/IoT infrastructures
- Pharmaceutical drug discovery
- Smart Grid energy forecasting

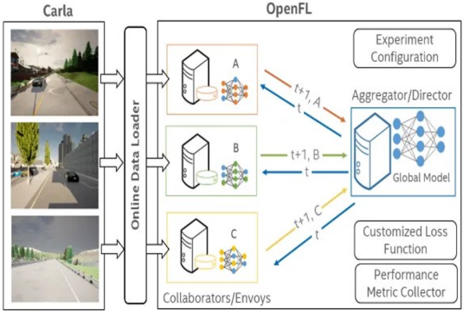
Overview

Applications of Federated Learning in Mobile Devices



- Can have millions of devices!
- Applications: fitness app, photography, next word prediction, Google voice etc.
- Less data in one device implies poor performance

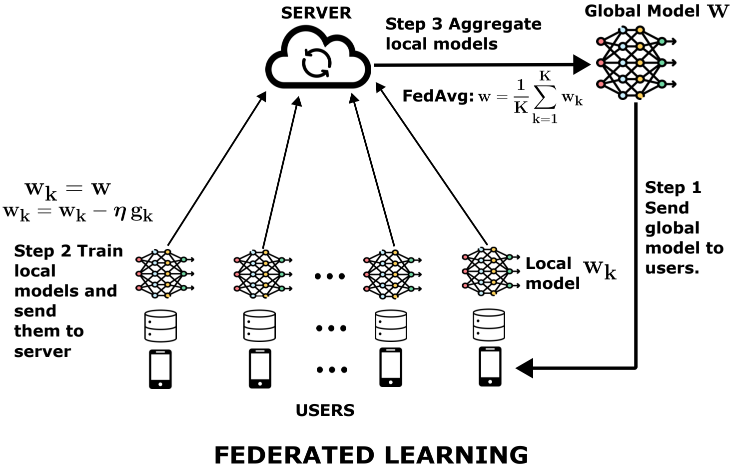
Autonomous driving (Object detection)



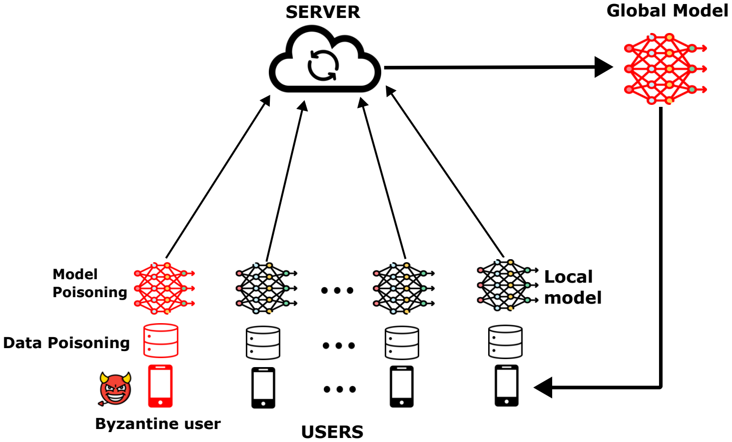
Outline

- **FL in Untrusted Environments**
  - Privacy leakage & Byzantine threats
  - Jointly handle privacy leakage and Byzantine threats?
- **Contributions**
  - Role of analog codes
  - Outlier detection with side information
- **FORTA Framework**
- **Discussion**

Federated Learning

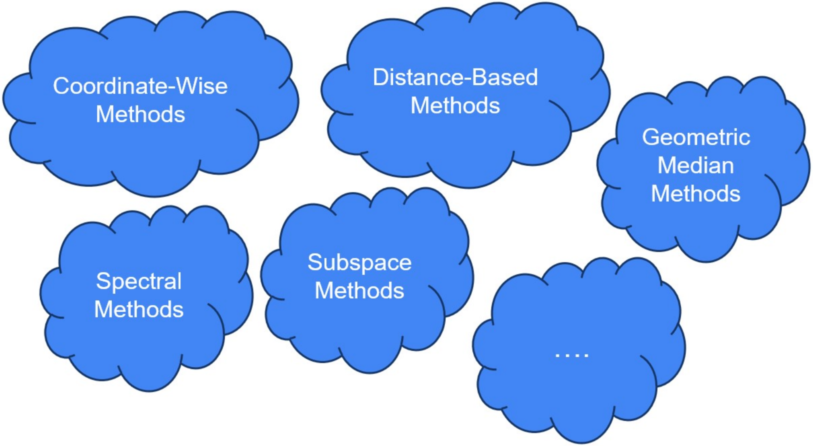


Model Poisoning in Federated Learning



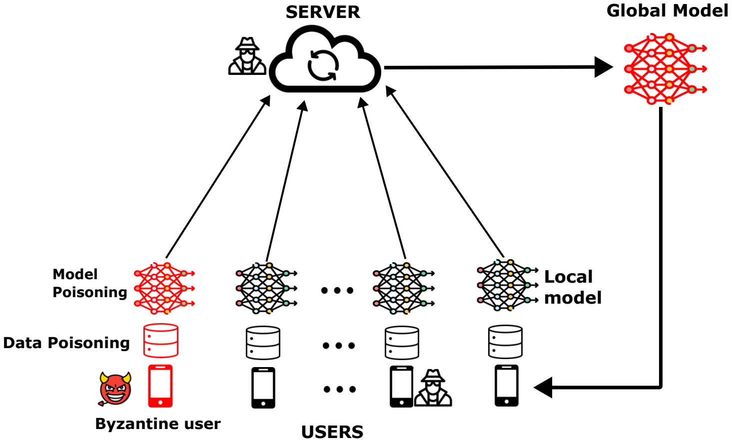
- A malicious user can send *corrupted updates*
- These updates can *corrupt the global model*
- Even a single Byzantine user may cause *large deviation*

How to Handle Byzantine Clients?



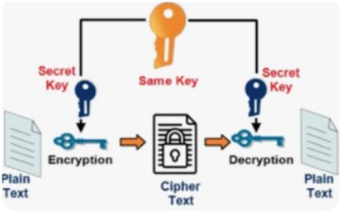
- Apply a suitable method to reject anomalies
- Aggregate the rest

Privacy Threat

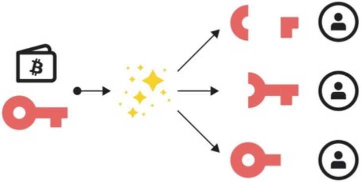


- **Gradient inversion attacks** [Zhu et al., 2019]: Recover training data.
- **Curious server**: Infer private data from user updates.
- **Colluding users**: Collaborate to try to leak the private data.

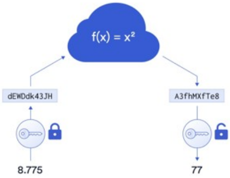
How to Handle Privacy Threat?



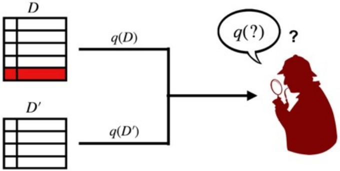
Cryptographic Primitives



Secret Sharing Methods



Homomorphic Encryption

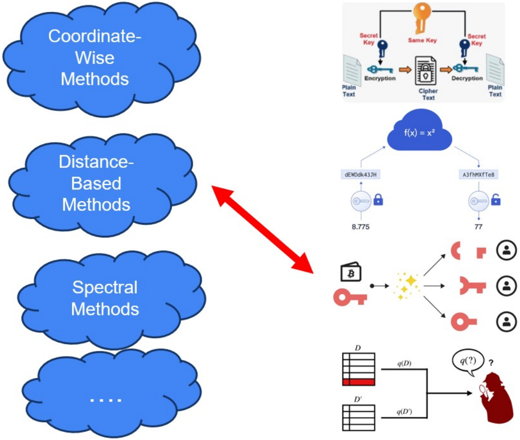


Differential Privacy

Motivation

Byzantine Resilience along with Privacy

- Integrating Krum rule with secret sharing methods

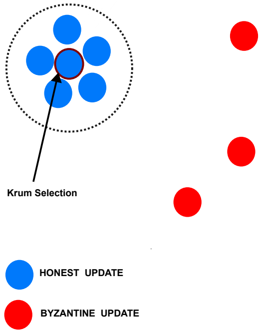


Revisiting Krum Rule [Blanchard et al., 2017]

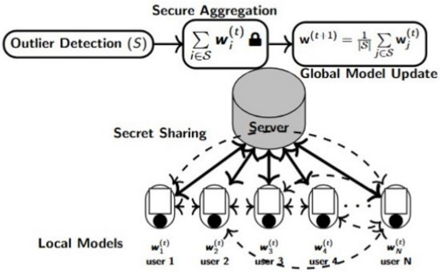
- Malicious updates arbitrary and unconstrained.
- **Krum rule:**
  - For each user i, compute total distance to its $N - f - 2$ **nearest neighbors**, denoted by $\mathcal{M}_i$.
  - Score:

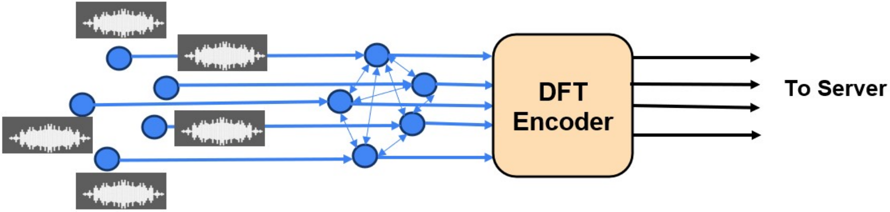
$$S_i = \sum_{j \in \mathcal{M}_i} \|\boldsymbol{w}_i - \boldsymbol{w}_j\|^2$$

  - Select the update with the smallest S_i.
- Requires access to pairwise distances $\|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2$.
- Krum selects the update most consistent with the majority.



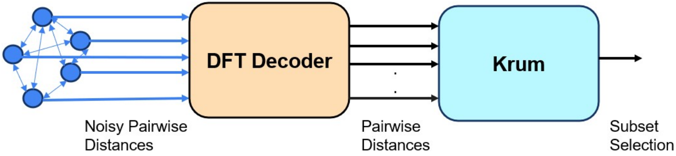
Contributions





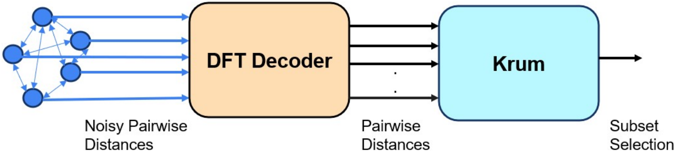
Contributions

Clients share noisy pairwise distances to the server

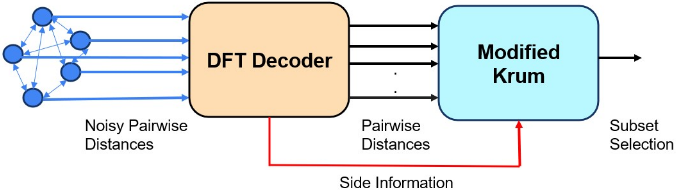


Contributions

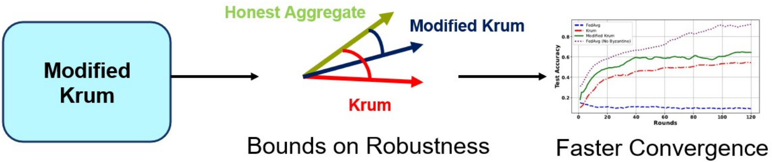
Clients share noisy pairwise distances to the server



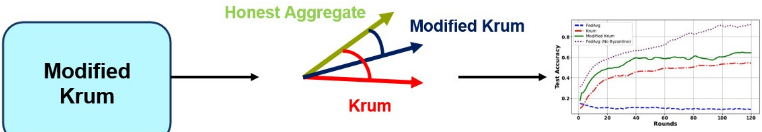
- Error-localization step introduces vulnerabilities
- Use decoder likelihood as prior



Contributions



Contributions



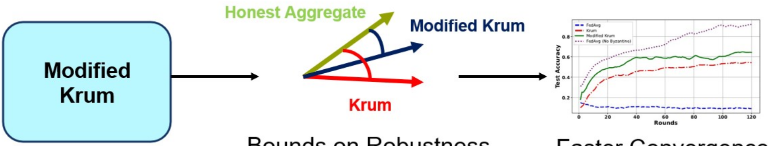
Bounds on Robustness

Faster Convergence

Theorem (Convergence Rate Scaling)

$$\zeta \in \mathcal{O}\left(\sqrt{\frac{\sigma^2}{T}\left(\frac{1}{N-f}+\mathcal{K}\right)}\right) + \mathcal{O}(\epsilon(N^2))$$

Contributions



Bounds on Robustness

Faster Convergence

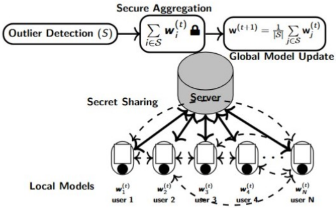
Theorem (Convergence Rate Scaling)

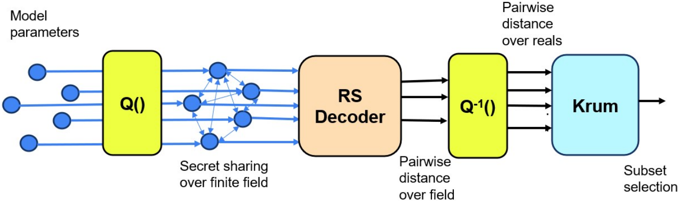
$$\zeta \in \mathcal{O}\left(\sqrt{\frac{\sigma^2}{T}\left(\frac{1}{N-f}+\mathcal{K}\right)}\right) + \mathcal{O}(\epsilon(N^2))$$

Key Result: $\mathcal{K}$ Factor Reduction

$$\mathcal{K} < \mathcal{K}_{\text{Krum}}$$

So et al., “Byzantine-resilient secure federated learning,” IEEE JSAC, 2021

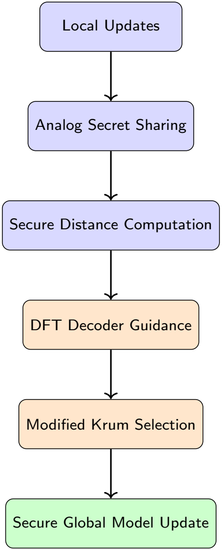




- Limitation: finite-field operations → computation overhead, overflows, dependence on field size.

FORTA Framework

- **FORTA: Fourier-Based Outlier-Resilient Trust Aggregation**
  - Secure aggregation framework operating entirely in the real domain
  - Combines analog secret sharing with Krum-based outlier detection
- **Principles:**
  - Preserving the privacy of users
  - Enables secure aggregation under malicious behavior
  - Mitigating finite-precision vulnerabilities



Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.

Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.
- User i encodes update $\boldsymbol{w}_i \in \mathbb{R}^d$ as a noisy polynomial:

$$\boldsymbol{P}_i(x) = \boldsymbol{w}_i + \sum_{j=1}^{T} \boldsymbol{r}_{ij} x^j + \boldsymbol{\epsilon}_i$$

$\boldsymbol{r}_{ij} \sim \mathcal{N}(0, \sigma_n^2 / T)$, $\boldsymbol{\epsilon}_i$ models precision loss, σ_p^2.

Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.
- User i encodes update $\boldsymbol{w}_i \in \mathbb{R}^d$ as a noisy polynomial:

$$\boldsymbol{P}_i(x) = \boldsymbol{w}_i + \sum_{j=1}^{T} \boldsymbol{r}_{ij} x^j + \boldsymbol{\epsilon}_i$$

  $\boldsymbol{r}_{ij} \sim \mathcal{N}(0, \sigma_n^2/T)$, $\boldsymbol{\epsilon}_i$ models precision loss, σ_p^2.
- Server and users agree on N-th roots of unity $\{\omega_1, \ldots, \omega_N\}$.

Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.
- User i encodes update $\boldsymbol{w}_i \in \mathbb{R}^d$ as a noisy polynomial:

$$\boldsymbol{P}_i(x) = \boldsymbol{w}_i + \sum_{j=1}^{T} \boldsymbol{r}_{ij} x^j + \boldsymbol{\epsilon}_i$$

  $\boldsymbol{r}_{ij} \sim \mathcal{N}(0, \sigma_n^2/T)$, $\boldsymbol{\epsilon}_i$ models precision loss, σ_p^2.
- Server and users agree on N-th roots of unity $\{\omega_1, \ldots, \omega_N\}$.
- User i computes shares at these points:

$$\boldsymbol{s}_{ij} = \boldsymbol{P}_i(\omega_j)$$

Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.
- User i encodes update $\boldsymbol{w}_i \in \mathbb{R}^d$ as a noisy polynomial:

$$\boldsymbol{P}_i(x) = \boldsymbol{w}_i + \sum_{j=1}^{T} \boldsymbol{r}_{ij} x^j + \boldsymbol{\epsilon}_i$$

  $\boldsymbol{r}_{ij} \sim \mathcal{N}(0, \sigma_n^2 / T)$, $\boldsymbol{\epsilon}_i$ models precision loss, σ_p^2.
- Server and users agree on N-th roots of unity $\{\omega_1, \ldots, \omega_N\}$.
- User i computes shares at these points:

$$\boldsymbol{s}_{ij} = \boldsymbol{P}_i(\omega_j)$$

- User i sends $\boldsymbol{s}_{ij}$ to user j.

Secret Sharing Phase

- Setup: N users, at most f Byzantine, and up to T may collude.
- User i encodes update $\boldsymbol{w}_i \in \mathbb{R}^d$ as a noisy polynomial:

$$\boldsymbol{P}_i(x) = \boldsymbol{w}_i + \sum_{j=1}^{T} \boldsymbol{r}_{ij} x^j + \boldsymbol{\epsilon}_i$$

  $\boldsymbol{r}_{ij} \sim \mathcal{N}(0, \sigma_n^2/T)$, $\boldsymbol{\epsilon}_i$ models precision loss, σ_p^2.
- Server and users agree on N-th roots of unity $\{\omega_1, \ldots, \omega_N\}$.
- User i computes shares at these points:

$$\boldsymbol{s}_{ij} = \boldsymbol{P}_i(\omega_j)$$

- User i sends $\boldsymbol{s}_{ij}$ to user j.
- Up to T colluding users learn nothing about $\boldsymbol{w}_i$

Pairwise Differences of Shares

- User i computes
$$d^i_{jk} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$

Pairwise Differences of Shares

- User i computes

$$d^i_{jk} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$

- Users send $\{d^i_{jk}\}$ to the server.

Pairwise Differences of Shares

- User i computes

$$d_{jk}^{i} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$

- Users send $\{d_{jk}^{i}\}$ to the server.
- Each d_{jk}^{i} is the evaluation of $\boldsymbol{h}_{jk}(x) := \|\boldsymbol{P}_j(x) - \boldsymbol{P}_k(x)\|^2$ at $x = \omega_i$.

Pairwise Differences of Shares

- User i computes

$$d^i_{jk} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$

- Users send $\{d^i_{jk}\}$ to the server.
- Each d^i_{jk} is the evaluation of $\boldsymbol{h}_{jk}(x) := \|\boldsymbol{P}_j(x) - \boldsymbol{P}_k(x)\|^2$ at $x = \omega_i$.
- Byzantine users may corrupt some evaluations.

Pairwise Differences of Shares

- User i computes
$$d^i_{jk} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$
- Users send $\{d^i_{jk}\}$ to the server.
- Each d^i_{jk} is the evaluation of $\boldsymbol{h}_{jk}(x) := \|\boldsymbol{P}_j(x) - \boldsymbol{P}_k(x)\|^2$ at $x = \omega_i$.
- Byzantine users may corrupt some evaluations.
- For a given pair j, k, the server collects N evaluations of $h_{jk}(x)$
$$\boldsymbol{c}_{jk} = \big(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\big).$$

Pairwise Differences of Shares

- User i computes
$$d^i_{jk} = \|\boldsymbol{s}_{ji} - \boldsymbol{s}_{ki}\|^2$$
- Users send $\{d^i_{jk}\}$ to the server.
- Each d^i_{jk} is the evaluation of $\boldsymbol{h}_{jk}(x) := \|\boldsymbol{P}_j(x) - \boldsymbol{P}_k(x)\|^2$ at $x = \omega_i$.
- Byzantine users may corrupt some evaluations.
- For a given pair j, k, the server collects N evaluations of $h_{jk}(x)$
$$\boldsymbol{c}_{jk} = \big(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\big).$$
- Server collects $\binom{N}{2}$ such vectors

DFT-Based Recovery at Server

- For each pair (j,k), the server receives a corrupted version of

$$\boldsymbol{c}_{jk} = \big(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\big)$$

  which is a noisy codeword of an $(N, 2T{+}1)$ DFT code.

DFT-Based Recovery at Server

- For each pair (j, k), the server receives a corrupted version of

$$\boldsymbol{c}_{jk} = \left(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\right)$$

  which is a noisy codeword of an $(N, 2T{+}1)$ DFT code.

-

$$\begin{bmatrix} 1 & \omega_1 & \cdots & \omega_1^{2T} \\ 1 & \omega_2 & \cdots & \omega_2^{2T} \\ 1 & \ldots & \ldots & \\ 1 & \omega_N & \cdots & \omega_N^{2T} \end{bmatrix} \begin{bmatrix} \|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2 \\ * \\ \vdots \\ * \end{bmatrix} + \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}$$

DFT-Based Recovery at Server

- For each pair (j, k), the server receives a corrupted version of

$$\boldsymbol{c}_{jk} = \left(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\right)$$

  which is a noisy codeword of an $(N, 2T{+}1)$ DFT code.

- $$\begin{bmatrix} 1 & \omega_1 & \cdots & \omega_1^{2T} \\ 1 & \omega_2 & \cdots & \omega_2^{2T} \\ 1 & \ldots & \ldots & \\ 1 & \omega_N & \cdots & \omega_N^{2T} \end{bmatrix} \begin{bmatrix} \|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2 \\ * \\ \vdots \\ * \end{bmatrix} + \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}$$

- Server applies a DFT decoder to correct errors and recover coefficients of $h_{jk}(x)$

DFT-Based Recovery at Server

- For each pair (j,k), the server receives a corrupted version of

$$\boldsymbol{c}_{jk} = \left(h_{jk}(\omega_1), \ldots, h_{jk}(\omega_N)\right)$$

  which is a noisy codeword of an $(N, 2T{+}1)$ DFT code.

- $$\begin{bmatrix} 1 & \omega_1 & \cdots & \omega_1^{2T} \\ 1 & \omega_2 & \cdots & \omega_2^{2T} \\ 1 & \ldots & \ldots & \\ 1 & \omega_N & \cdots & \omega_N^{2T} \end{bmatrix} \begin{bmatrix} \|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2 \\ * \\ \vdots \\ * \end{bmatrix} + \begin{bmatrix} * \\ * \\ \vdots \\ * \end{bmatrix}$$

- Server applies a DFT decoder to correct errors and recover coefficients of $h_{jk}(x)$
- Server recovers

$$\|\boldsymbol{w}_j - \boldsymbol{w}_k\|^2 + \zeta_{jk}$$

Observations on Analog Codes

Finite Precision & Subtle Attacks

- Finite-precision arithmetic introduces noise in local evaluations.
- Error localization becomes unreliable.
- Byzantine users exploit by crafting perturbations to evade error localisation.
- Distances corrupted.

Observations on Analog Codes

Finite Precision & Subtle Attacks

- Finite-precision arithmetic introduces noise in local evaluations.
- Error localization becomes unreliable.
- Byzantine users exploit by crafting perturbations to evade error localisation.
- Distances corrupted.

Our Approach

- Decode across all $\binom{N}{2}$ codewords, not independently.
- Exploit the fact that adversarial corruptions are consistent across codewords.
- Improves error localization.

Joint Localization Strategy

1. Received Codewords

↓

2. Syndrome Computation
DFT Parity-check matrix

↓

3. Error-locator
$\Lambda(x)$ construction

↓

4. Custom Chien Search
Pick f smallest eval of
$|\Lambda(x)|^2$ over roots of unity

↓

5. GMM Clustering
Byzantine vs Precision

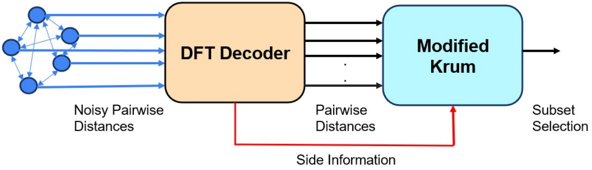
↓

6. Error locations
Top f in index frequency profile

↓

7. Final Correction
$\widetilde{c}_{jk} = c_{jk} - \tilde{e}$ and
compute $\hat{h_{jk}}(0)$ for $\hat{d_{jk}}$

Outlier Detection with DFT-Guided Krum



- **From Frequency to Trust**
  - Decoder builds frequency profile: f_i = how often user i is flagged.
  - Convert to confidence weights via softmax:

$$p_i = \frac{\exp(f_i/\tau)}{\sum_{j=1}^{N} \exp(f_j/\tau)}$$

- **Modified Krum Score**

$$S_i^{\text{mod}} = p_i S_i + (1 - p_i) S_{\min}, \quad S_{\min} = \frac{\min_i S_i}{N - f - 1}$$

Selection & Secure Aggregation

- **Selection:** Server picks m users with lowest modified scores.
  - Ensures aggregation favors consistent & trusted updates.

- **Secure Aggregation:**
  - Server broadcasts trusted set $\mathcal{S}$.
  - Each user $i \in \mathcal{S}$ sends masked share $\boldsymbol{s}_i = \sum_{j \in \mathcal{S}} \boldsymbol{s}_{ji}$.
  - Server reconstructs

$$\sum_{j \in \mathcal{S}} \mathbf{w}_j$$

    via DFT-based decoding.
  - Final update:

$$\boldsymbol{w}^{(t+1)} = \frac{1}{|\mathcal{S}|} \sum_{j \in \mathcal{S}} \mathbf{w}_j^{(t)}$$

Theoretical Framework: Resilient Averaging

We adopt the standard notion of (f, λ)-resilience, Farhadkhani et al. 2022

Definition

For $f < N$ and $\lambda \geq 0$, an aggregation rule F is said to be (f, λ)-*resilient* if for any collection of vectors $x_1, \ldots, x_N$ and any honest set $S \subseteq \{1, \ldots, N\}$ of size $N - f$,

$$\left\| F(x_1, \ldots, x_N) - \bar{x}_S \right\| \;\leq\; \lambda \max_{i,j \in S} \|x_i - x_j\|$$

where $\bar{x}_S$ is the average of honest vectors.

Interpretation

- Measures the deviation between the aggregated update from the true honest mean.
- Lower $\lambda \implies$ tighter bound $\implies$ stronger resilience.

Resilient Averaging of Decoder-Guided Krum

Geometric Bound

On a high-probability reconstruction event, the Decoder-Guided Modified Multi-Krum is (f, Λ)-resilient, where:

$$\left\| F(x_1, \ldots, x_N) - \bar{x}_S \right\| \;\leq\; \lambda \max_{i,j \in S} \|x_i - x_j\| + 2\sqrt{\epsilon}$$

such that

$$\lambda \propto \left(1 + \sqrt{\frac{1 + (N - f - 1) R_{\mathrm{mod}}}{N - 2f}} \right)$$

Insights:

- λ is not **static**. It depends on the confidence ratio R_{mod} provided by the DFT decoder.
- Strong decoder feedback implies tighter bound

Convergence Guarantees

Theorem

Under standard smoothness assumptions, with learning rate $\gamma \propto 1/\sqrt{T}$, the expected gradient norm satisfies:

$$\mathbb{E}[\|\nabla \mathcal{L}(\widehat{\theta})\|^2] \;\leq\; \underbrace{\mathcal{O}\left(\sqrt{\frac{\sigma^2}{T}\left(\mathbb{E}[\lambda^2](N-f)+\frac{1}{N-f}\right)}\right)}_{\textit{Robust Convergence Term}} \;+\; \underbrace{\mathcal{O}(\epsilon(N^2))}_{\textit{Decoder Noise Floor}}$$

Sketch:

- Compute bound on $\mathbb{E}[\left\|F(x_1,\ldots,x_N)-\bar{x}_S\right\|^2]$
- Plug the bound in the quadratic error term in smoothness [Farhadkhani et al. 2022]

Convergence Rate Analysis

Theorem (Convergence Rate Scaling)

$$\zeta \in \mathcal{O}\left(\sqrt{\frac{\sigma^2}{T}\left(\frac{1}{N-f} + \mathcal{K}\right)}\right) \; + \; \mathcal{O}\big(\epsilon(N^2)\big)$$

where

$$\mathcal{K} = \left(1 + \sqrt{\frac{1 + (N-f-1)\mu}{N-2f}}\right)^2 (N-f)$$

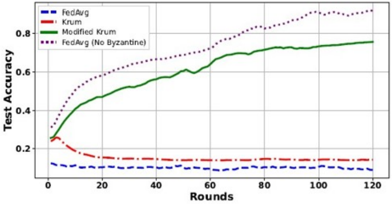
- Special case of Krum when $\mu = 1$

$$\mathcal{K}_{\text{krum}} = \left(1 + \sqrt{\frac{N-f}{N-2f}}\right)^2 (N-f)$$

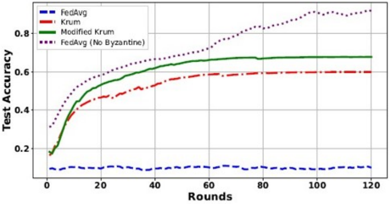
- Significant improvement over Krum when $\mu \to 0$

$$\mathcal{K} = \left(1 + \sqrt{\frac{1}{N-2f}}\right)^2 (N-f)$$

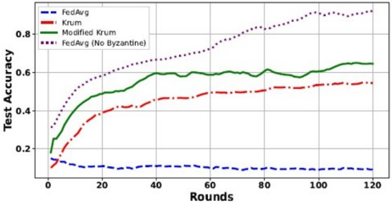
Experimental Results



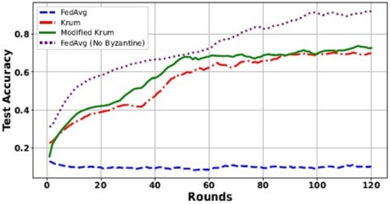
Empire attack



Min-Max attack

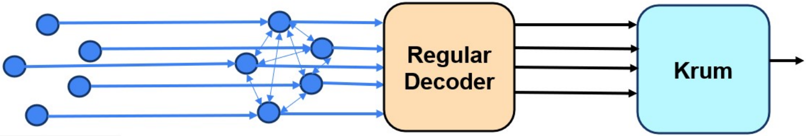


LIE attack



Sign-flip attack

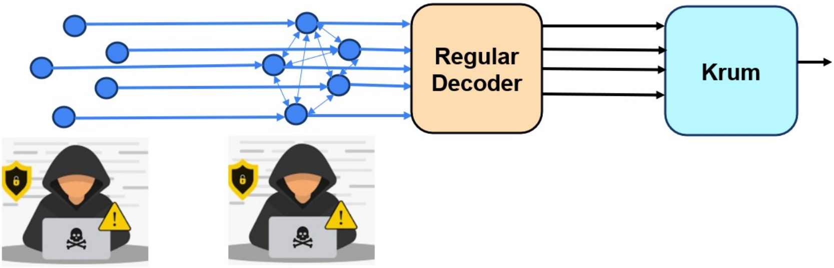
Big Picture



Performance:

- As good as vanilla Krum
- precision noise effect

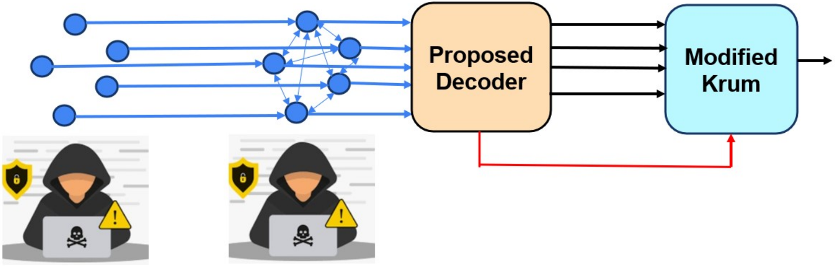
Big Picture



Performance:

- Worse than vanilla Krum
- precision noise effect

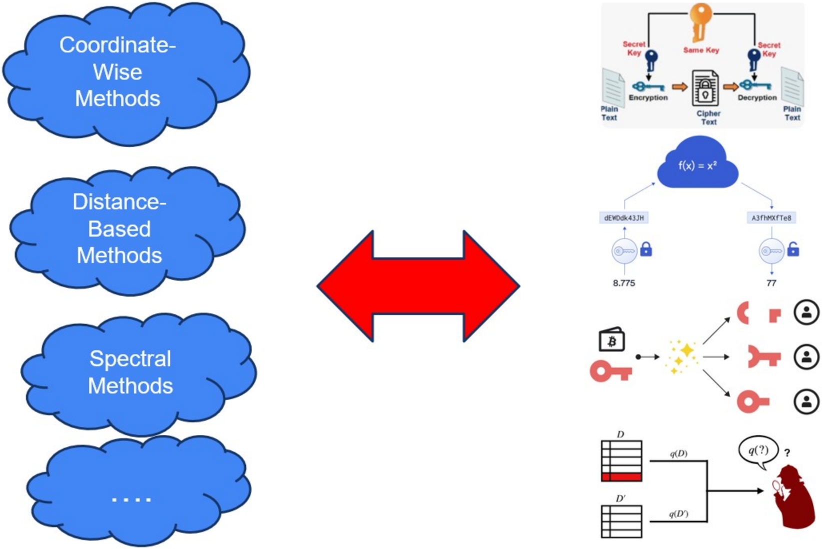
Big Picture



Performance:

- Better than vanilla Krum
- Negligible precision noise effect

Discussion


Coordinate-Wise Methods
Distance-Based Methods
Spectral Methods
....
Secret Key
Same Key
Secret Key
Encryption
Decryption
Plain Text
Cipher Text
Plain Text
f(x) = x²
dEWDdk43JH
A3fhMXfTe8
8.775
77
D
q(D)
D′
q(D′)
q(?)
?