FLEDGE: Ledger-based Federated Learning Resilient to Inference and Backdoor Attacks

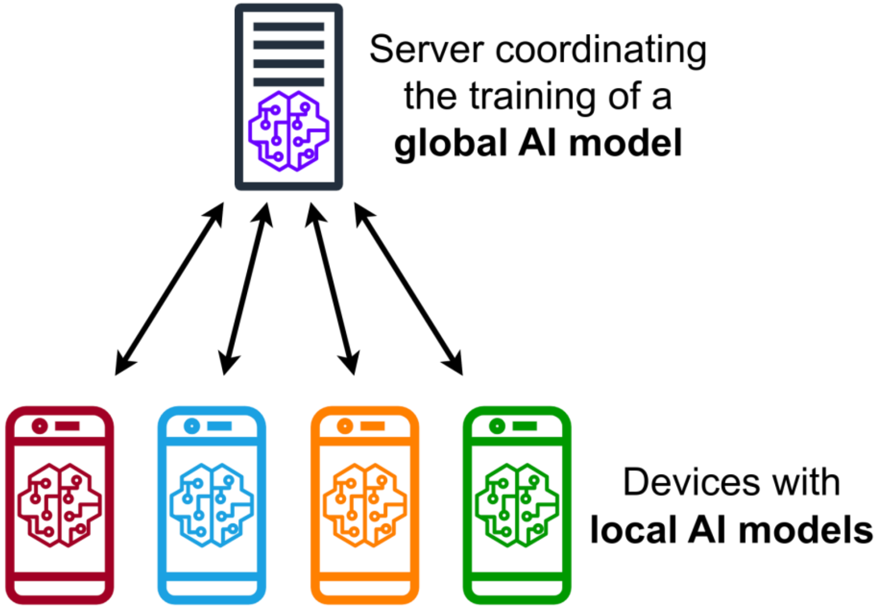
J. CASTILLO[1], P. RIEGER[3], H. FEREIDOONI[3], Q. CHEN[2], A. SADEGHI[3]

1 The University of Texas Rio Grande Valley

2 UTSA Roadrunners + 3 Technische Universität Darmstadt

Problem Statement

- Machine Learning (ML) is *very popular* for different applications
  - Problem: Data collection is difficult due to security and privacy concerns
  - Solution: Federated Learning (FL)
- FL aims to solve the privacy concerns by distributing the learning process
  - Clients train model with local private data – local model
  - Aggregation server compiles a new model using all local models – global model
- Privacy Problem in FL
  - You must trust the aggregation server
  - Consequences: Adversary can analyze local models to retrieve private data from clients
- Security Problem in FL
  - You must trust the learning process
  - Consequences: Adversary can poison data and/or model to skew the learning process



Generic Image from Wikipedia

Adversary Model

- Privacy Threat
  - White-box Inference Attack – *honest-but-curious*
  - Goal: Adversary extracts sensitive information from every local model before aggregation
  - Capabilities: Full control of aggregation server
- Security Threat
  - Targeted Poisoning Attack – *backdoors*
  - Goal: Adversary manipulates loss function to train models to behave normally all the time except when a specific set of conditions, e.g., trigger, is present in the input
  - Capabilities: Control of $\boldsymbol{f}$ clients out of $\boldsymbol{n}$ total clients such that $f = \frac{n}{2}$

Existing Solutions

- Privacy-preserving Defenses
  - Secure Multi-Party Computation (SMPC) based Solutions
  - Multi-party Homomorphic Encryption Solution
- Poisoning Defenses
  - Untargeted Poisoning Solutions
  - Backdoor Solutions
- Hybrid Defenses
  - SMPC + Poisoning Solutions
  - TEE + Poisoning Solutions
- Research Gap – Lack of accountability
  - Malicious aggregation service
  - Malicious training clients

Requirements

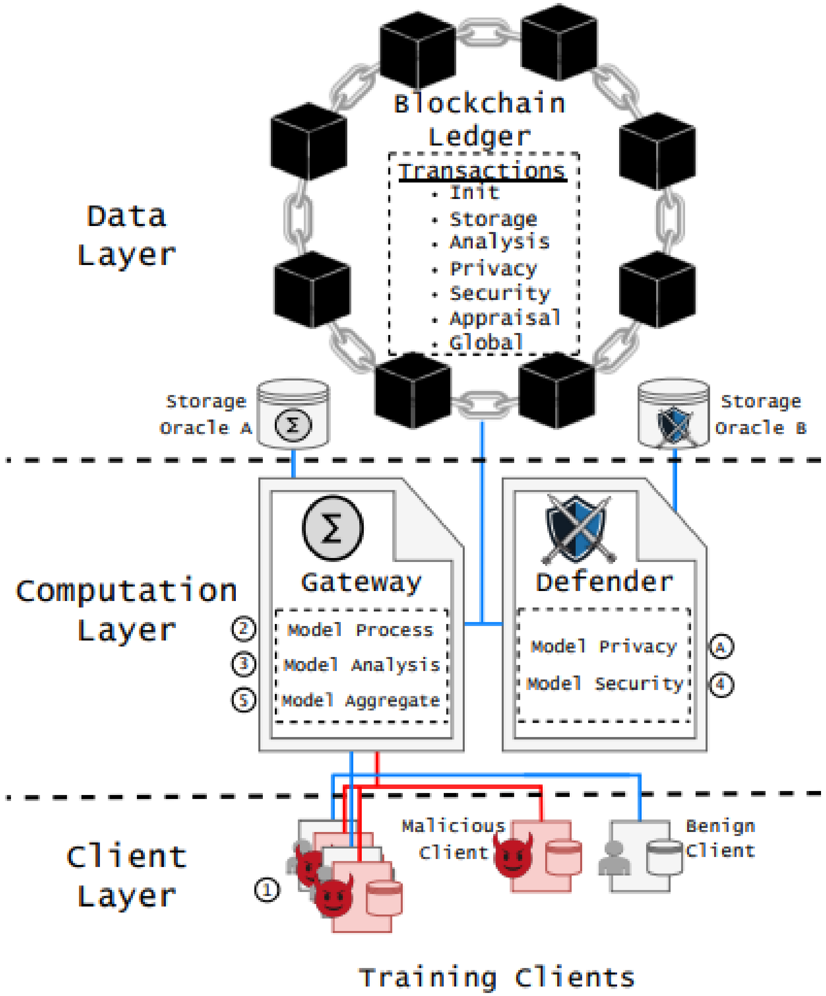
- P1: Utility Retention
  - Defense must preserve model utility
- P2: Computation Availability
  - Private model analysis and aggregation shall not fail due to limited resource availability
- S1: Effective Poisoning Mitigation
  - Defense must detect poisoning attempts
  - Defense must mitigate their impact on the global model
  - Defense must preserve model utility
- S2: Autonomous Behavior
  - Defense must be flexible to automatically adjust to different adversarial strategies

Challenges

- C1: Leverage Blockchain to improve trust between computation parties
- C2: Combine Homomorphic Encryption and Blockchain to limit the ledger's transparency
- C3: Solve the dilemma of preventing the server from analyzing the local models against inference attacks while having to inspect the local models to detect poisoned models
- C4: Discriminate poisoned models to prompt disciplinary actions
- C5: Credit clients over training rounds to make malicious clients accountable for their attacks

Proposed Solution: FLEDGE

- Contributions
  1. Strong privacy guarantees via Blockchain Two-Contract Computation (BT2C)
     - BT2C – Semi-honest relationship between 2 smart contracts using CKKS
     - Resilient against white-box inference attacks
  2. Mitigation of poisoning attacks via G-KDE clustering
     - Evaluated on 4 datasets: MNIST, Fashion-MNIST, CIFAR10 and Reddit
  3. Compensation algorithm via **cryptocurrency**
     - Offer incentives to benign aggregation services and benign training clients



Transaction Name	Transaction Type
Init	TT1
Storage	TT2
Analysis	TT3
Privacy	TT4
Security	TT5
Appraisal	TT6
Global	TT7

Assumptions

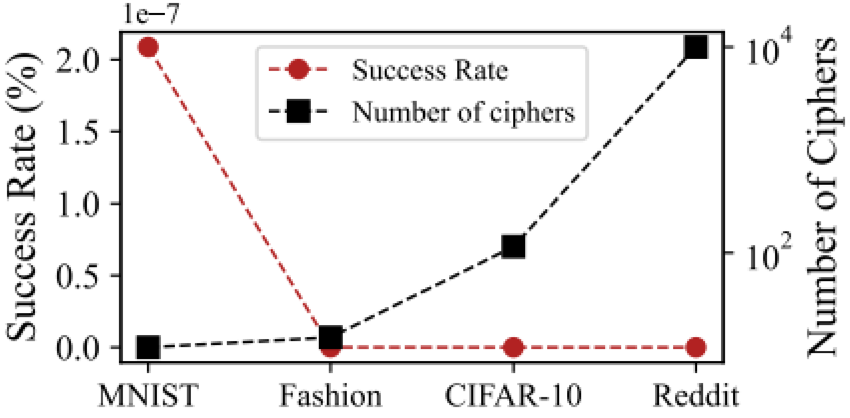
- A1: Consensus Protocol is NOT compromised
  - Blockchain is the platform of our solution
  - We rely on default consensus protocol – Raft
- A2: Non-colluding Servers
  - Servers engage in semi-honest relationship to enable privacy
  - Adversary cannot control both servers simultaneously
- A3: Clients Perform Encryption – CKKS
  - Clients are summed to have sufficient computational resources to perform encryption

Workflow

- Step 0 : Initialization
  - Interested party (owner) proposes learning task of T training rounds with reward R for the training session
  - Owner submits global model parameters (TT7) to be trained
  - Owner submits TT1 to start training session
- Step 1: Model Encryption
  - Client i trains model W_i using private (local) data
  - Client i injects noise δ_i to offset W_i s.t. $W'_i = W_i + \delta_i$
  - Client i encrypts W'_i and δ_i (W_i^* and δ_i^*, respectively) and submits them to GWC
- Step 2: Model Process
  - GWC stores W_i^* into storage oracle A and generates model ID
  - GWC uses model ID, client ID and δ_i^* to submit TT2
- Step 3: Model Analysis
  - GWC uses δ_i^* to offset G_{t-1}^* s.t. $G_{t-1}^* = G_{t-1}^* + \delta_i^*$
  - GWC computes cosine distance c_i between W_i^* and G_{t-1}^* using DC as computation party – BT2C: Private Cosine Distance (Alg. 1)
  - GWC uses c_i and model ID to submit TT3
- Step A: Model Privacy – BT2C: Secure Decryption (Alg. 2)
  - DC checks if GWC is attempting to misbehave – TT4
  - DC adjusts reward for GWC to remove malicious intent
- Step 4: Model Security
  - DC applies Poisoning Defense (Alg. 3) to remove malicious models
  - DC adjusts rewards for training clients to remove malicious intent
  - DC uses model IDs and rewards to submit TT5 and TT6, respectively
- Step 5: Model Aggregate
  - GWC use filtered models to compute new global model G_t and G_t^* – BT2C: Private Aggregation (Alg. 4)
  - GWC uses new models to submit TT7

Evaluation: Inference Attacks

Application	IC			WP
Datasets	MNIST	Fashion	CIFAR-10	Reddit
#Records	70K	70K	60K	20.6M
Model	CNN	CNN	$\text{ConvMixer}_{256/3}$	LSTM
#params	$\sim$ 23K	$\sim$ 29K	$\sim$ 234K	$\sim$ 20M
#ciphers	12	15	115	$\sim$ 10.1K

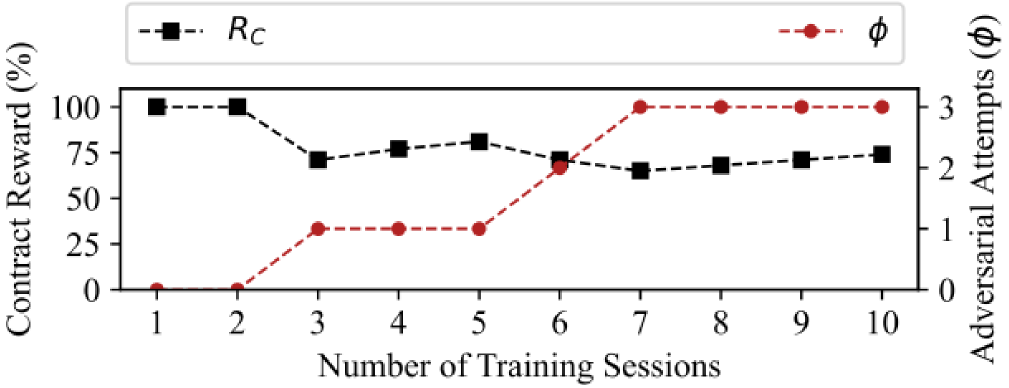


Evaluation: Poisoning Attacks

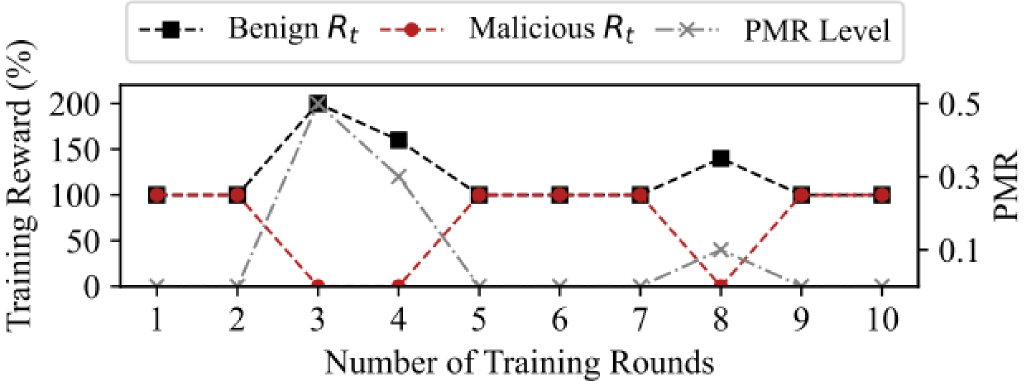
Poisoning Attack	Dataset	No Defense		FLEDGE	
		BA	MA	BA	MA
Untargeted	Reddit	–	15.8	–	22.7
	MNIST	–	91.5	–	98.3
	Fashion	–	41.1	–	90.0
	CIFAR-10	–	28.9	–	83.0
Constrain-and-Scale	Reddit	100	22.6	0.0	22.7
	MNIST	98.0	87.7	0.4	98.3
	Fashion	100.0	69.3	2.4	90.6
	CIFAR-10	100.0	66.1	0.0	83.8
DBA	Reddit	100.0	22.6	0.0	22.7
	MNIST	82.6	77.2	0.1	98.3
	Fashion	99.7	36.7	1.0	98.3
	CIFAR-10	85.2	67.4	2.1	83.8

Defenses	Reddit		MNIST		Fashion		CIFAR-10	
	BA	MA	BA	MA	BA	MA	BA	MA
Benign Setting	0.0	22.7	0.5	98.3	3.7	90.9	0	83.9
No Defense	100.0	22.7	98.0	87.7	100.0	69.2	100.0	66.1
Krum	100.0	22.6	0.6	**98.3**	2.8	90.1	**0.0**	83.0
FoolsGold	**0.0**	**22.7**	0.5	**98.3**	3.0	90.7	**0.0**	83.6
Auror	100.0	22.5	0.5	**98.3**	2.5	**90.9**	**0.0**	**83.9**
AFA	100.0	22.6	83.1	94.2	97.9	87.3	100.0	66.5
DP	77.0	22.0	26.5	97.3	52.2	88.6	60.0	76.6
FLEDGE	**0.0**	**22.7**	**0.4**	**98.3**	**2.4**	90.6	**0.0**	83.8

Evaluation: Reward System



(a) R_C behavior under ϕ



(b) R_t behavior under PMR

$$R_C = 0.1 * R * e^{-(\phi+1)/s}$$

ϕ = # of anomalies = # of TT4

s = # of sessions = # of TT1

$$R_T = \frac{R - R_C}{T * len(g_b)}$$

g_b = benign group of models

Limitations and Future Work

- Limitations
  - Storage Costs – Homomorphic Encryption
  - Computation Costs – Blockchain and Homomorphic Encryption
  - Reward System is connected to Defense's performance
- Future Work
  - In-depth analysis into scalability – transaction fees, communication costs
  - Performance analysis based on different blockchain platforms

Summary

- **C1:** Leverage Blockchain to improve trust between computation parties
  - Blockchain Two-Contract Computation – BT2C
- **C2:** Combine Homomorphic Encryption and Blockchain to limit the ledger's transparency
  - Use of noise constant (δ). An attacker would need to break each delta to learn model's parameters
- **C3:** Solve the dilemma of preventing the server from analyzing the local models against inference attacks while having to inspect the local models to detect poisoned models
  - BT2C – Private Cosine Distance + Private Aggregation
- **C4:** Discriminate poisoned models to prompt disciplinary actions
  - G-KDE Poisoning Defense
- **C5:** Credit clients over training rounds to make malicious clients accountable for their attacks
  - Reward System

Blockchain Two-Contract Computation – BT2C

Algorithm 1: BT2C – Private Cosine Distance

Input: δ^* ◂ encrypted offset
G^* ◂ encrypted global model
W^* ◂ encrypted local model

$Z_D \leftarrow$ PrivateDotProduct($G^* + \delta^*, W^*$)
$X_D \leftarrow$ SecureDecryption(Z_D) ◂ defender function
$Z_G \leftarrow$ PrivateMagnitudeSquared($G^* + \delta^*$)
$X_G \leftarrow$ SecureDecryption(Z_G)
$Z_L \leftarrow$ PrivateMagnitudeSquared(W^*)
$X_L \leftarrow$ SecureDecryption(Z_L)
$c \leftarrow 1 - \frac{\sum_{i=1}^{n} X_{D_i}}{\sqrt{\sum_{i=1}^{n} X_{G_i}} * \sqrt{\sum_{i=1}^{n} X_{L_i}}}$
UpdateScoreToLedger(c) ◂ new TT3

Algorithm 2: BT2C – Secure Decryption

Input : $z_1, \ldots, z_m$ ◂ computation ciphers
Output : X ◂ array of decrypted numbers
ρ ◂ array of decrypted model chunks

$\delta_1^*, \ldots, \delta_K^* \leftarrow$ ReadOffsetFromLedger() ◂ from TT2
$S_k \leftarrow$ ReadKeyFromStorage()
$t \leftarrow 0.05$ ◂ array variation tolerance
for *each cipher i in [1, m]* **do**
  $\rho_i \leftarrow$ Decrypt(z_i, S_k)
  $v \leftarrow \left| \frac{\max(\rho_i) - \min(\rho_i)}{\max(\rho_i)} \right|$ ◂ compute variation
  if $v \leq t$ **then**
    $X_i \leftarrow$ Average(ρ_i)
  else if $K > 1$ **then**
    for *each offset j in [1, K]* **do**
      $\delta_j \leftarrow$ Decrypt(δ_j^*, S_k)
    end
    $\rho_i \leftarrow \frac{\rho_i - \sum_{j=1}^{K} \delta_j}{K}$ ◂ offset removal/injection
  else
    $R \leftarrow$ ReadRewardFromLedger() ◂ from TT1
    $s \leftarrow$ CountSessionsFromLedger() ◂ # TT1
    $\phi \leftarrow$ CountAnomaliesFromLedger() ◂ # TT4
    $R_C \leftarrow 0.1 * R * e^{-(\phi+1)/s}$ ◂ calculating reward
    UpdateContractRewardToLedger(R_C) ◂ new TT4
    $\rho_i \leftarrow \emptyset$ ◂ empty set
  end
end
return X *or* p ◂ output type dependent on process

Algorithm 4: BT2C – Private Aggregation

Input: $W_1^*, \ldots, W_N^*$ ◂ selected models

$Z \leftarrow W_1^*$ ◂ encrypted base model
for *each update i in [2, N]* **do**
  $Z \leftarrow$ Add(Z, W_i^*)
end
$G_t \leftarrow$ SecureDecryption(Z) ◂ defender function
$P_k \leftarrow$ ReadKeyFromLedger() ◂ from TT1
$G_t^* \leftarrow$ Encrypt(G_t, P_k)
UpdateGlobalToLedger(G_t^*, G_t) ◂ new TT7

G-KDE Poisoning Defense

Algorithm 3: Poisoning Defense

Input: $(c_i, \ldots, c_K)$ ◃ distance scores

$f \leftarrow 2000$ ◃ resolution factor for smooth curves

$(x_1, \ldots, x_f), (y_1, \ldots, y_f) \leftarrow \mathrm{GaussianKDE}([c_i, \ldots, c_K], f)$ ◃ compute gaussian kernel density estimation

$(l_1, \ldots, l_N) \leftarrow \mathrm{LocalMinimums}([y_1, \ldots, y_f])$ ◃ l_n is the index of local minimum found in y

$G \leftarrow \{[x_1, x_{l_1}], \ldots, [x_{l_{N-1}}, x_{l_N}], [x_{l_N}, x_f]\}$ ◃ group set based on local minimums

$M \leftarrow N + 1$ ◃ maximum number of available groups

for *each group* m *in* $[1, M]$ **do**

    for *each score* i *in* $[1, K]$ **do**

        if $c_i \in G_m$ **then**

            $g_m \leftarrow i$ ◃ append model index i to a group

        end

    end

end

$\mathrm{UpdateGroupsToLedger}(g)$ ◃ new TT5. g_1 is closest to G_{t-1}

$R \leftarrow \mathrm{ReadRewardFromLedger}()$ ◃ from TT1

$T \leftarrow \mathrm{ReadTotalNumberOfRoundsFromLedger}()$ ◃ from TT1

$R_C \leftarrow \mathrm{ReadContractRewardFromLedger}()$ ◃ from TT4

$R_\tau \leftarrow \frac{R - R_C}{T * \mathrm{len}(g_1)}$ ◃ training reward

$\mathrm{UpdateTrainingRewardToLedger}(R_\tau)$ ◃ new TT6