Federated Graph Intelligence for Financial Fraud Detection

A unified, privacy-preserving graph learning platform to combat financial fraud and illicit finance.

Temitope Adeyeha
02-01-2025

Table of Contents

1. Executive Summary

Financial fraud, cyber-crime and illicit-funds movement increasingly operate as **networked schemes**. Fraud rings coordinate across multiple accounts, counterparties and institutions, yet most detection remains siloed within individual banks or money-service providers. The U.S. Department of the Treasury's Financial Crimes Enforcement Network (FinCEN) recognizes cybercrime and fraud among the most significant AML/CFT threats[1] (FinCEN.gov), and the Bank Secrecy Act (BSA) manual describes Suspicious Activity Reporting (SAR) as the **cornerstone** of the regulatory framework for combating money-laundering[2] (ffiec.gov).

Existing fraud/AML systems rely heavily on **static rules** or narrow feature spaces and are hard to adapt when adversaries change tactics. Graph neural networks (GNNs) can capture complex patterns (fan-in mule collectors, layering chains, smurfing bursts) that are difficult to represent in tabular transaction data[3] (Graph Neural Networks for Financial Fraud Detection: A Review, 2024). However, many financial institutions cannot share data directly because of privacy laws and competitive concerns. Federated learning (FL) lets institutions collaborate by exchanging **model updates** rather than raw data. In FL each institution trains locally and a server averages the updates; the FedAvg algorithm reduces communication rounds and can handle unbalanced, non-IID data[4] (McMahan). To preserve confidentiality, secure aggregation protocols ensure the server learns only the sum of updates[5] (Bonawitz).

This white paper specifies a deployable technical design called **Federated Graph Intelligence (FGI)**. Each financial institution builds a *person-to-person* transaction graph and trains a **temporal graph neural encoder** locally. Institutions send signed, encrypted model updates to a consortium aggregator, which performs robust aggregation and returns an improved global model. The design includes layered defenses against model poisoning and privacy attacks, a sentinel test suite to detect anomalies, and a compliance harness aligned with regulatory expectations. By learning jointly without sharing customer data, FGI aims to improve fraud detection rates, reduce false positives and support investigators in writing SAR narratives.

2. Industry Background

2.1 Regulatory Context

The U.S. AML/CFT framework stresses that **fraud and cybercrime** are priority threats[1] (FinCEN.gov). Banks and money-service businesses must file Suspicious Activity Reports; the BSA/AML manual states that SARs "form the **cornerstone of the BSA reporting system**" and are critical for combating money-laundering and other crimes[2] (ffiec.gov). Institutions are expected to have policies, procedures, staffing, and periodic testing to support detection and reporting[6] (ffiec.gov). In parallel, the **NIST AI Risk Management Framework** lists characteristics of trustworthy AI—valid and reliable, safe, secure and resilient, accountable and transparent, explainable and interpretable, privacy-enhanced and fair[7] ((AIRC))—while the **NIST Cybersecurity Framework 2.0** organizes risk management into six functions (Govern, Identify, Protect, Detect, Respond and Recover)[8] (NIST). A federated system designed for fraud detection must therefore address privacy, safety, explainability and governance.

2.2 Limitations of Current Systems

Most fraud detection platforms operate within a single institution and use **rule-based** systems or gradient-boosted models built on transaction-level features. These systems struggle to detect **cross-institutional patterns** and adapt slowly to new fraud typologies. Financial transaction graphs are dynamic, high-dimensional and often non-IID; static models cannot capture complex network interactions. Additionally, institutions are hesitant to share data because of privacy laws and competition; collaboration is limited to informal information sharing.

2.3 Opportunities with Graph Learning and Federated Collaboration

Graph neural networks (GNNs) learn embeddings that capture relational and temporal patterns, outperforming traditional methods on fraud and money-laundering detection[3] (Graph Neural Networks for Financial Fraud Detection: A Review, 2024). However, training a centralized graph model would require pooling customer data across institutions, raising privacy concerns. **Federated learning** addresses this by averaging local gradients; the FedAvg algorithm combines local stochastic gradient descent with server-side averaging to reduce communication and handle unbalanced data[4] (McMahan). Secure aggregation protocols allow the server to learn only the sum of updates[5] (Bonawitz), and robust aggregation methods like trimmed mean, median or geometric median defend against corrupted updates[9]. Combining GNNs with FL enables collaborative learning on distributed graph data while satisfying privacy and compliance requirements.

3. Problem Statement

3.1 Motivating Scenario

Fraud rings often operate across multiple banks, using mule collectors, layering chains and smurfing bursts to evade detection. When a group coordinates transactions across several institutions, no single bank sees enough of the pattern to detect the scheme. Current detection systems are local and rule-based; they cannot detect patterns that span two or more hops or differentiate benign bursts from orchestrated smurfing. As a result, illicit funds may move through the system for days before investigators notice.

3.2 Specific Challenges

- **Institution-siloed data:** Banks cannot share raw transaction data due to privacy laws and competition. Regulators encourage information sharing (e.g., Section 314(b) of the USA PATRIOT Act) but there is no interoperable, privacy-preserving infrastructure.
- **Complex network patterns:** Financial transaction graphs exhibit fan-in/fan-out structures, dynamic bursts and cycles. GNNs can capture such structures, but naive implementations oversmooth or require expensive memory modules. Dynamic graphs require temporal modelling.
- **Non-IID distributions:** Different institutions have different customer bases and transaction patterns. In federated settings, the same GNN architecture may perform differently across clients; results from FedGraphNN show that centralized models often lose their advantage under non-IID splits[10] (He, 2021). Schema mismatches (different feature definitions or windowing) can silently degrade performance.
- **Security and robustness:** The server cannot inspect local training due to secure aggregation; malicious participants could poison the model. Without layered defenses, federated learning is vulnerable to backdoors and model poisoning.
- **Operational latency:** Fraud interdiction requires real-time scoring (milliseconds). Graph computations and message passing can be heavy; the system must precompute embeddings and support incremental updates.

4. Proposed Solution: Federated Graph Intelligence

FGI is a privacy-preserving collaboration layer that enables financial institutions to learn shared fraud detection models without sharing customer data. The design includes a

graph formulation, **temporal graph neural encoder**, **federated training protocol**, **robust and secure aggregation**, and **operational blueprint** for deployment.

4.1 System Overview

1. **Local graph construction:** Each participating institution builds a *directed, temporal, weighted graph* where nodes represent persons (customer identities) and edges represent transfers. Edge attributes include amount, timestamp, channel/type (e.g., ACH, wire), and direction. The system intentionally excludes devices and accounts in this baseline to minimise heterogeneity; future versions can extend to heterogeneous graphs.
2. **Feature engineering:** Each node and edge is assigned feature vectors derived from local data (Section 4.2). Standardized units, aggregation windows and missing-value conventions ensure interoperability.
3. **Temporal graph encoder:** A shared backbone model encodes node and edge features into embeddings using **decayed message passing** and **direction-specific transformations** (Section 4.3).
4. **Local training:** Institutions train the encoder on their labeled cases and self-supervised objectives. They compute local updates for a fixed number of epochs.
5. **Secure aggregation:** Institutions encrypt and sign their updates. A central aggregator uses secure aggregation to compute the sum of updates without seeing individual contributions[5].
6. **Robust aggregation and conformance:** The server performs robust aggregation (trimmed mean/median or geometric median) to tolerate corrupted updates[9]. Conformance checks and sentinel tests verify that updates follow the schema and do not degrade performance. Trust weights adjust each participant's influence based on historical reliability.
7. **Model distribution and online scoring:** The improved global model is returned to participants. Each institution updates its local encoder, recomputes embeddings on sliding windows, and serves low-latency scores via a calibrated head. Near-real-time scoring uses precomputed "stable" embeddings and incremental "burst" updates.

4.2 Key Components

Node Feature Schema

Each node v is associated with a feature vector x_v. A recommended baseline includes:

- **Identity and lifecycle:** tenure buckets; KYC/verification tier; number of linked accounts; customer segment flags (consumer vs SMB); prior suspiciousness indicators.
- **Behavioral traffic:** counts of outgoing/incoming transactions in multiple windows (e.g., 1 hour, 24 hours, 7 days, 30 days); unique counterparties per window; entropy of counterparties; active-hour distribution; burstiness metrics.
- **Flow structure:** in-degree and out-degree per window; in/out totals; in/out ratio; concentration measures (Gini or Herfindahl indexes) over counterparties.
- **Statistical stability:** mean/variance of transaction amounts; log-amount quantiles; seasonality flags; change-point indicators.

These features align with typical "unusual activity" indicators used in bank monitoring programs and support SAR narrative development.

Edge Feature Schema

Each edge (u, v, t, a) (a transfer from sender u to receiver v at time t with amount a) is associated with features $e_{u,v}$:

- Normalized amount (e.g., z-score by sender); timestamp features (day of week, hour); time since last outgoing from u and last incoming to v.
- Channel/rail type (one-hot or embedding); optional flags for same-day vs scheduled transfers; rail type (business vs consumer).
- Direction-specific roles (incoming vs outgoing), encoded implicitly in the message functions.

Interoperability is critical: units are standardized (amounts in cents; timestamps in UTC millis), a fixed set of aggregation windows is used, and missingness is handled explicitly (mask vectors or sentinel values).

Optional Components

- **Heterogeneity:** Later versions can extend the graph to heterogeneous node types (accounts, devices, merchants). Heterogeneous Graph Transformer (HGT) models support type-dependent attention and temporal encoding[11].
- **Relation-specific variants:** The encoder can be replaced by a temporal R-GCN where each channel/rail is treated as a relation with its own weight matrix[12] (Hamilton, 2018).

- **Contrastive or infomax objectives:** Self-supervised augmentation (edge dropout, feature masking, time jitter) and graph contrastive learning can strengthen embeddings under distribution shift[12].

4.3 User Types

- **Consumers:** individuals transferring funds. Risk scores inform transaction monitoring and account-level interdiction.
- **Merchants / SMBs:** recipients or senders; risk scores feed into merchant onboarding and payment rail decisions.
- **Partners:** third-party payment processors or FinTechs integrating the model via an SDK.
- **Institutions:** banks and money-service businesses that join the federation. They run local training and scoring pipelines; some may adopt managed hosting models.

4.4 Federated Training Protocol

The FGI consortium uses a **synchronous federated learning** protocol based on *Federated Averaging (FedAvg)*[4]. Let there be K participating institutions and denote the global model parameters at round t by θ_t. Each institution i has a local dataset D_i of n_i labeled and unlabeled transactions and associated node/edge features. The goal is to solve a global risk minimization problem

$$\min_{\theta} \sum_{i=1}^{K} p_i \; \mathbb{E}_{x \sim D_i}[\mathcal{L}_i(\theta; x)],$$

where $p_i = n_i / \sum_j n_j$ is the institution's weight and $\mathcal{L}_i$ is the local loss (a mixture of supervised and self-supervised objectives defined below). A typical round proceeds as follows:

1. **Distribution of the global model:** The server sends the current parameters θ_t (and any hyper-parameters) to a selected subset S_t of institutions. All models are signed and hashed so that clients can verify authenticity.

2. **Local training:** Each institution performs E epochs of stochastic gradient descent on its local data to obtain updated parameters $\theta_t^{(i)}$. The local update is computed as the difference

$$\Delta\theta_t^{(i)} = \theta_t^{(i)} - \theta_t.$$

3. **Norm clipping and differential privacy:** Before transmitting, each client scales its update to bound its ℓ_2-norm. Given a clipping norm S, the update is clipped as

$$\Delta\theta_t^{(i)} \leftarrow \Delta\theta_t^{(i)} \cdot \min\left(1, \frac{S}{\|\Delta\theta_t^{(i)}\|_2}\right),$$

and optional client-level differential-privacy noise is added

$$\Delta\theta_t^{(i)} \leftarrow \Delta\theta_t^{(i)} + \mathcal{N}(0, \sigma^2 S^2 \mathbf{I}),$$

where σ controls the privacy–utility trade-off[13] (Geyer, 2018). Clipping and DP noise protect privacy and limit the influence of any single client[13].

4. **Secure aggregation:** Participants encrypt and sign their clipped updates; a secure aggregation protocol computes the sum of updates $\sum_{i\in S_t} \Delta\theta_t^{(i)}$ without revealing any individual contribution[5]. The protocol tolerates dropouts and supports high-dimensional vectors[14].

5. **Robust aggregation and trust weighting:** The server computes the weighted average of updates to obtain the new model

$$\theta_{t+1} = \theta_t + \sum_{i\in S_t} w_i \, \Delta\theta_t^{(i)},$$

where $w_i = p_i \times \tau_i$ combines the institution's data weight p_i and a **trust weight** $\tau_i \in [0,1]$ reflecting its historical reliability. Instead of naive averaging, the aggregator may apply coordinate-wise median, trimmed mean or geometric median to the set $\{\Delta\theta_t^{(i)}\}_{i\in S_t}$ to defend against corrupted updates[9]. The aggregate update is then normalized by the sum of weights.

6. **Round outcome and rollback:** The aggregated model undergoes conformance and sentinel tests. If tests pass, θ_{t+1} is released to participants. Otherwise, the round is **rejected** and the latest good model is retained; offending participants may be quarantined. This ensures that poisoning or anomalous updates do not propagate.

This protocol enables collaborative learning across institutions while protecting privacy, limiting the impact of adversaries and accommodating non-IID data distributions[4].

5. Technical Architecture

5.1 System Architecture

FGI adopts a **five-stage pipeline** for each institution: (1) **Graph builder** ingests transaction streams and updates the temporal graph in sliding windows; (2) **Feature store** computes and versions node and edge features; (3) **Local GNN trainer** runs local training with labeled and self-supervised objectives; (4) **Embedding store** persists node embeddings for low-latency retrieval; (5) **Online scorer** uses precomputed embeddings and a small calibrated head to score candidate transactions in milliseconds. Stable embeddings are refreshed hourly/daily, while burst embeddings update within minutes for active nodes.

5.2 Data Layer

Each institution maintains a local graph $G_i = (V_i, E_i)$ and data D_i derived from transactions. Graph windows (e.g., 1 day, 7 days, 30 days) are used to support both short-term and long-term behaviors. Features are computed deterministically and logged for auditing. Embeddings are stored in a key-value store keyed by person ID; float16 embeddings (128 dimensions) require ~256 bytes per node, enabling tens of millions of active nodes.

5.3 AI / Machine Learning Layer

Temporal Decayed Message Passing

The encoder learns hidden states $h_v^{(k)}$ for each node v in layer k. Let N_v^+ be inbound neighbors and N_v^- outbound neighbors. For each edge (u, v) with timestamp t we compute an edge embedding $\phi(e_{u,v})$ and a **time-decay factor**:

$$\lambda_{u,v} = \exp\Big(- lpha\,(T - t)\Big),$$

where T is the inference time and lpha is a learned or fixed decay parameter. Messages are computed separately for inbound and outbound edges using small MLPs or linear maps ψ_{extin} and ψ_{extout}:

$$m_{u\to v}^{\text{in}} = \lambda_{u,v}\,\psi_{\text{in}}\Big(h_u^{(k-1)},\,\phi(e_{u,v})\Big)\,,$$

$$m_{v\to u}^{\text{out}} = \lambda_{v,u}\,\psi_{\text{out}}\Big(h_v^{(k-1)},\,\phi(e_{v,u})\Big)\,.$$

Messages are aggregated with a permutation-invariant operator, e.g., **degree-normalized mean**:

$$\overline{m}_v = rac1|N_v^+| \sum_{u \in N_v^+} m^{extin}_{uightarrowv} + rac1|N_v^-| \sum_{u \in N_v^-} m^{extout}_{vightarrowu}.$$

The node state is updated via a nonlinear transformation (e.g., ReLU) and optional layer normalization or dropout:

$$h_v^{(k)} = (W^{(k)} [h_v^{(k-1)} \| \overline{m}_v] ig),$$

where $W^{(k)}$ is a learnable weight matrix. Two layers ($k = 2$) typically suffice to propagate information across 1–2 hops while maintaining interpretability; deeper layers risk over-smoothing and increase latency[12].

Output Heads

- **Entity risk:** After the encoder, a small MLP produces a probability p_v that node v belongs to the "fraudulent" class:

$$p_v = (\mathrm{MLP}_e xtnode(h_v) ig).$$

- **Transaction risk:** For a candidate transaction (u, v) with real-time features $f_{u,v}$, compute edge score:

$$p_{u,v} = (\mathrm{MLP}_e xtedge(h_u \| h_v \| f_{u,v}) ig).$$

This "encoder + small task-specific head" structure decouples the shared representation from institution-specific risk tolerances.

Training Objectives

FGI employs a mixed objective because fraud labels are sparse and delayed. Let $\hat{y}_v$ denote the predicted probability for node v, with ground-truth label $y_v \in \{0,1\}$. The **binary cross-entropy (BCE) loss** for node classification is

$$\mathcal{L}_{\mathrm{BCE}}^{\mathrm{node}} = -\sum_{v \in \mathcal{V}_{\mathrm{lab}}} [\, y_v \log \hat{y}_v + (1 - y_v) \log(1 - \hat{y}_v)],$$

where $\mathcal{V}_{\mathrm{lab}}$ is the set of labeled nodes. A similar BCE loss $\mathcal{L}_{\mathrm{BCE}}^{\mathrm{edge}}$ applies to labeled edges. Self-supervised objectives encourage temporal consistency and representation quality. The overall local loss for institution i is a weighted sum

$$\mathcal{L}_i(\theta) = \mathcal{L}_{\mathrm{BCE}}^{\mathrm{node}} + \beta\, \mathcal{L}_{\mathrm{BCE}}^{\mathrm{edge}} + \gamma\, \mathcal{L}_{\mathrm{temp}} + \eta\, \mathcal{L}_{\mathrm{contrast}} + \rho\ \| \theta \|_2^2,$$

where $\mathcal{L}_{\mathrm{temp}}$ is the temporal edge-prediction loss, $\mathcal{L}_{\mathrm{contrast}}$ is the contrastive or Infomax loss, and $\beta, \gamma, \eta, \rho$ are hyper-parameters controlling the contribution of each term. These

weights are tuned to balance supervised and self-supervised objectives. $\| \theta \|_2^2$ is an ℓ_2 regularizer that mitigates overfitting.

In summary, the mixed objective combines supervised classification, temporal prediction and representation learning. Parameter serialization uses canonical tensor ordering to enable deterministic hashing and signatures; 32-bit weights are used for training, with optional 16-bit quantized encodings for transmission.

- **Supervised node classification:** Binary cross-entropy loss over labeled nodes (confirmed fraud cases).
- **Supervised edge scoring:** Optional cross-entropy loss over labeled suspicious transactions.
- **Self-supervised temporal edge prediction:** Sample positive edges (observed events) and negatives (non-edges); train the model to predict whether an edge exists at time t (akin to Temporal Graph Networks).
- **Graph contrastive learning:** Generate augmented views of the graph via edge dropout, feature masking or time jitter; maximize agreement between embeddings of the same node under different views while minimizing agreement with negative samples.
- **Optional Infomax objective:** Maximize mutual information between local node embeddings and a global summary to encourage globally coherent representations.

Loss terms are weighed to balance supervised and self-supervised objectives. Parameter serialization uses canonical tensor ordering to enable deterministic hashing and signatures; 32-bit weights are used for training, with optional 16-bit quantized encodings for transmission.

5.4 Security & Privacy

Secure Aggregation

Federated learning aggregates local updates without revealing individual updates. **Secure aggregation** protocols allow the server to compute the sum of encrypted updates and learn only the aggregate[5]. The protocols tolerate up to one-third client dropouts and support high-dimensional vectors[14]. In FGI, institutions encrypt and sign update payloads; the aggregator computes the weighted sum and normalizes by client sample sizes.

Robust Aggregation and Differential Privacy

Naive averaging is vulnerable to **Byzantine participants** that submit corrupted updates. Robust aggregation replaces averaging with primitives like coordinate-wise median or trimmed mean, or uses algorithms such as Krum or geometric median; these methods are resilient to corrupted updates[9]. Each institution also **clips** its update to a maximum ℓ_2-norm: given a clipping norm S, the raw update $\Delta\theta$ is rescaled as

$$\Delta\theta \leftarrow \Delta\theta \ \min\left(1, \frac{S}{\|\Delta\theta\|_2}\right).$$

To protect privacy, optional **client-level differential privacy** noise is added

$$\Delta\theta \leftarrow \Delta\theta + \mathcal{N}(0, \sigma^2 S^2 \mathbf{I}),$$

where σ controls the noise level[13]. These defenses limit the impact of adversarial updates and protect the privacy of clients. After clipping and noise addition, a secure aggregation protocol computes the sum of the masked updates without revealing individual contributions[5]; the server then applies a robust aggregator (median, trimmed mean, Krum, geometric median) to the decrypted sum.[9]

Conformance Checks and Sentinel Evaluation

Federated systems are vulnerable because the aggregator cannot see local training. FGI implements a **compliance harness**: model artifacts are signed; clients verify signatures; the server verifies update envelopes (tensor shapes, embedding dimensions, dtype). Schema conformance and model contract checks ensure participants adhere to the agreed feature schema and embedding sizes. A **sentinel test suite** runs canary patterns (fan-in mule collector, fan-out burst, layering chains, cycles, smurfing micro-payments) and negative controls to detect if the aggregated model behaves unusually. If sentinel metrics degrade or anomalies occur, the round is rolled back and offending participants are quarantined. Trust scores for participants are updated based on conformance, update stability and sentinel outcomes.

Five-Layer Anti-Poisoning Stack

1. **Participant governance:** Permissioned membership with contractual onboarding, cryptographic client identities and rate limits; minimum quorum per round.
2. **Clipping and DP noise:** Norm clipping and optional client-level differential privacy reduce the influence and precision of malicious updates[13].
3. **Robust aggregation:** Use median, trimmed mean, Krum or geometric median to aggregate updates[9].

4. **Pre-aggregation anomaly screening:** Require signed auxiliary statistics (loss deltas, norms, cosine similarity) even under secure aggregation; reject updates with abnormal statistics.
5. **Post-aggregation sentinel tests and rollback:** Treat the aggregated model as untrusted until it passes sentinel evaluation; rollback and quarantine participants if tests fail.

These layers provide defense-in-depth: failure of one control does not imply catastrophic compromise.

6. Operational Workflow

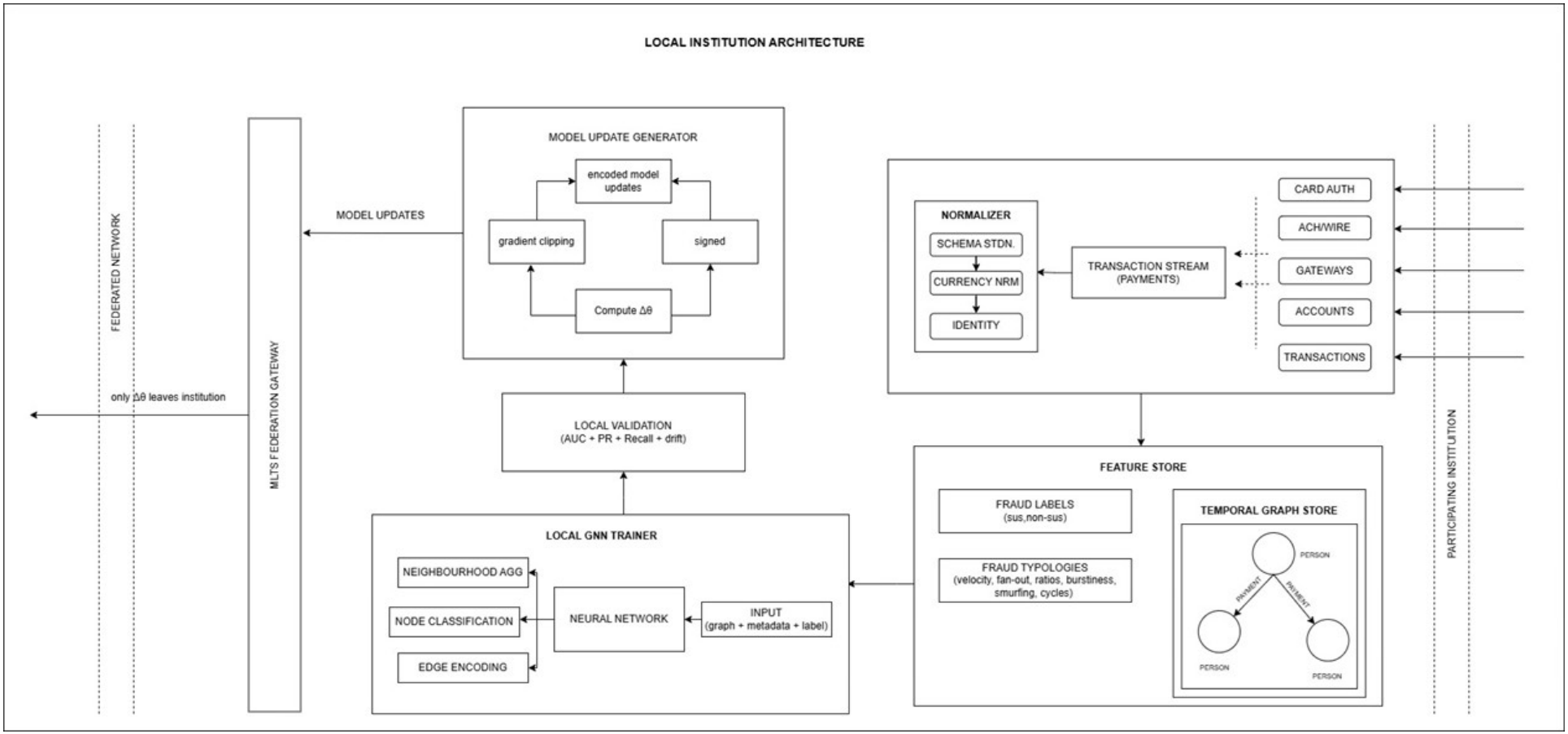
6.1 System Architecture

FGI comprises multiple institution-local pipelines and a central **federation network**:

1. **Institution-Local Pipeline:** Each institution ingests a transaction stream into a **graph builder**, which sends data to a **feature store**. A **local GNN trainer** updates the model, stores embeddings and serves an online scoring API. Model updates are prepared and transmitted to the federation.

2. **Federation Network:** Receives signed updates from participants, runs secure aggregation and robust averaging (e.g., coordinate-wise trimmed mean, median, Krum or geometric-median rules), executes sentinel tests on the aggregated model, and either publishes it to a model registry or rolls back if tests fail. The network also maintains a threat-intelligence store with known typologies and canary patterns.

Below are conceptual diagrams of the institution-local pipeline and the federation network for FGI.


LOCAL INSTITUTION ARCHITECTURE
FEDERATED NETWORK
MLTS FEDERATION GATEWAY
only Δθ leaves institution
MODEL UPDATES
MODEL UPDATE GENERATOR
encoded model updates
gradient clipping
signed
Compute Δθ
LOCAL VALIDATION
(AUC + PR + Recall + drift)
LOCAL GNN TRAINER
NEIGHBOURHOOD AGG
NODE CLASSIFICATION
EDGE ENCODING
NEURAL NETWORK
INPUT
(graph + metadata + label)
NORMALIZER
SCHEMA STDN.
CURRENCY NRM
IDENTITY
TRANSACTION STREAM
(PAYMENTS)
CARD AUTH
ACH/WIRE
GATEWAYS
ACCOUNTS
TRANSACTIONS
FEATURE STORE
FRAUD LABELS
(sus,non-sus)
FRAUD TYPOLOGIES
(velocity, fan-out, ratios, burstiness, smurfing, cycles)
TEMPORAL GRAPH STORE
PERSON
PAYMENT
PAYMENT
PERSON
PERSON
PARTICIPATING INSTITUTION

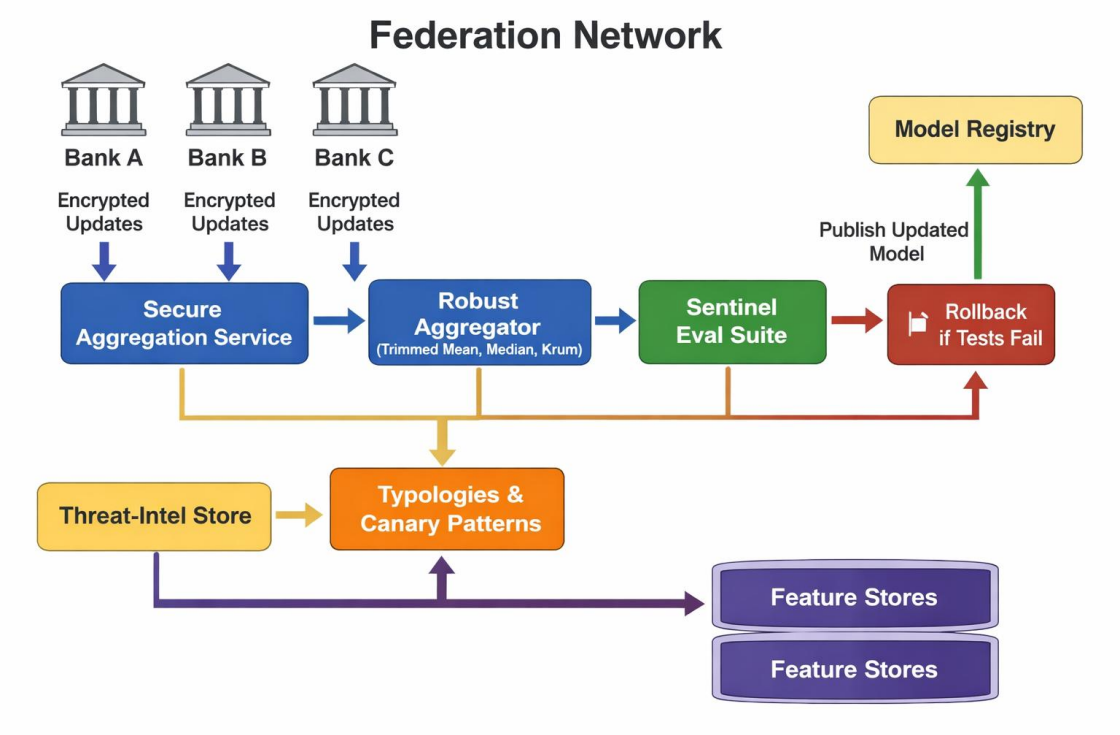
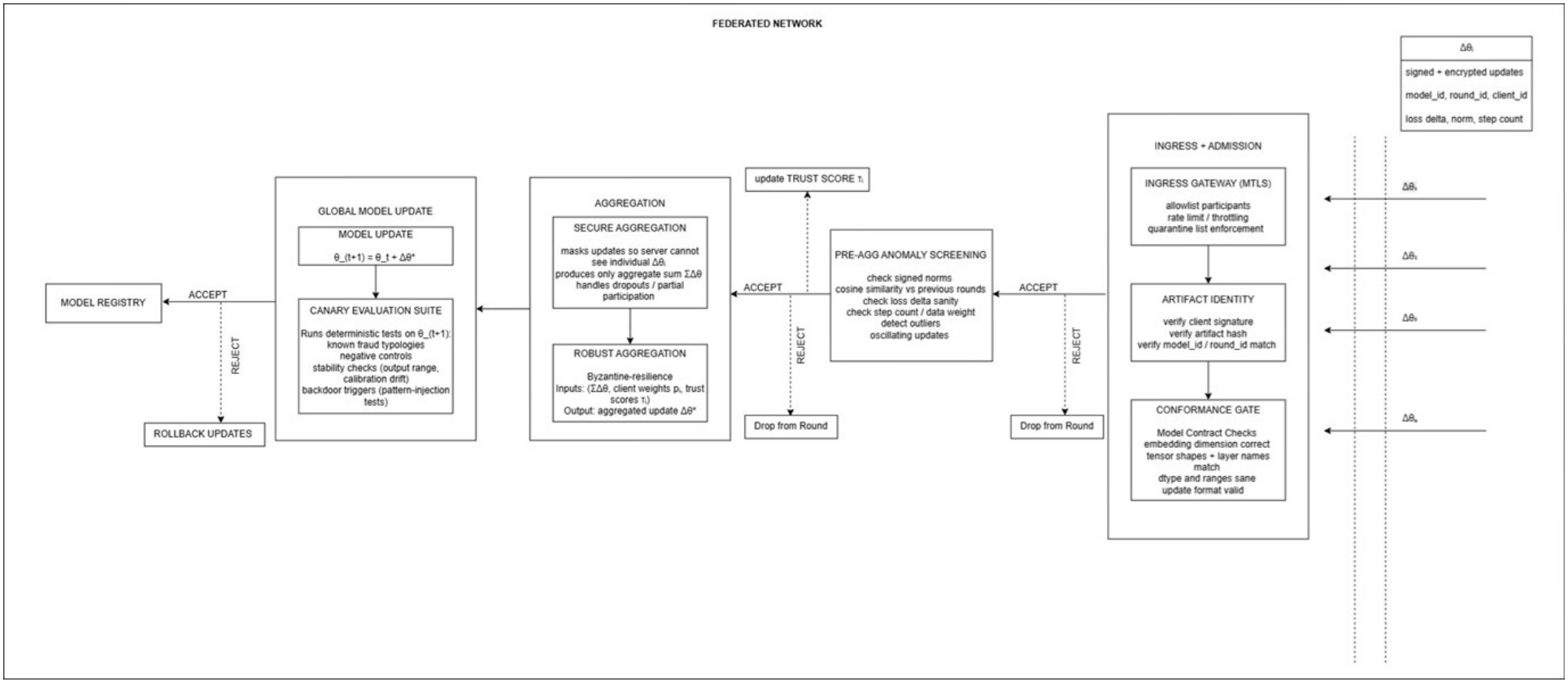


Fig 1: Pictorial representation of federation network showing contribution framework


FEDERATED NETWORK
Δθᵢ
signed + encrypted updates
model_id, round_id, client_id
loss delta, norm, step count
INGRESS + ADMISSION
INGRESS GATEWAY (MTLS)
allowlist participants
rate limit / throttling
quarantine list enforcement
ARTIFACT IDENTITY
verify client signature
verify artifact hash
verify model_id / round_id match
CONFORMANCE GATE
Model Contract Checks
embedding dimension correct
tensor shapes + layer names match
dtype and ranges sane
update format valid
ACCEPT
REJECT
Drop from Round
PRE-AGG ANOMALY SCREENING
check signed norms
cosine similarity vs previous rounds
check loss delta sanity
check step count / data weight
detect outliers
oscillating updates
update TRUST SCORE τᵢ
AGGREGATION
SECURE AGGREGATION
masks updates so server cannot see individual Δθᵢ
produces only aggregate sum ΣΔθ
handles dropouts / partial participation
ROBUST AGGREGATION
Byzantine-resilience
Inputs: (ΣΔθ, client weights pᵢ, trust scores τᵢ)
Output: aggregated update Δθ*
GLOBAL MODEL UPDATE
MODEL UPDATE
θ_(t+1) = θ_t + Δθ*
CANARY EVALUATION SUITE
Runs deterministic tests on θ_(t+1):
known fraud typologies
negative controls
stability checks (output range, calibration drift)
backdoor triggers (pattern-injection tests)
MODEL REGISTRY
ROLLBACK UPDATES

6.2 Data Layer

FGI relies on a **standard schema** to ensure interoperability:

- **Nodes:** Represent persons, each with features such as tenure, verification tier, number of linked accounts, transaction counts across windows, unique counterparties, burstiness, in-flow/out-flow ratios and stability statistics (mean/variance, quantiles, change-point indicators).
- **Edges:** Represent transactions with features including amount, timestamp, channel type (ACH, wire, P2P rail), normalised amount (sender-centric z-score), inter-event times and optional flags (e.g., same-day vs scheduled).
- **Windowing:** Features are computed over fixed windows (e.g., 1 hour, 24 hours, 7 days, 30 days) and amounts are expressed in a consistent unit (cents) in UTC.

This standardisation mitigates the risk of **schema mismatches** that plagues federated graph learning and ensures that all participants' updates contribute meaningfully to the global model.

6.3 End-to-End Flow

1. **Customer transaction:** A customer initiates a payment.
2. **Graph update:** The institution's graph builder records the transaction, updating node and edge features in sliding windows.
3. **Embedding refresh:** Stable embeddings are updated hourly/daily; burst embeddings for active nodes update in minutes.
4. **Online scoring:** For each new transaction (u, v), the online scorer retrieves embeddings h_u and h_v, computes a candidate score $p_{u,v}$ with a calibrated head, and uses thresholding or ranking to decide whether to flag or allow the transaction.
5. **Case management:** Suspicious transactions or entities trigger internal investigations and possible SAR filings.
6. **Local training:** Periodically (e.g., daily or multiple times per week) the institution trains the encoder on newly labeled fraud cases plus self-supervised objectives to reduce label sparsity.
7. **Update preparation:** The local client clips the update, adds optional DP noise and signs the payload.
8. **Federated round:** The consortium orchestrates a round; participating institutions send encrypted updates; the aggregator performs secure and robust aggregation;

sentinel tests run; if the round passes, the aggregated model is distributed back to participants.

9. **Deployment and monitoring:** Institutions deserialize the new model, verify signatures and update their pipelines; metrics (AUC, precision-recall, alert volume, false positives, typology recall, drift tests) are tracked and reported for model governance.

6.4 Cold-Start Handling

When a new customer or account has insufficient graph history, FGI falls back to a **cold-start score** based on local node features only. The system prioritizes embedding computation for the new node and its neighborhood.

6.5 Deployment Options

FGI supports three deployment models:

- **Local (on-prem/VPC):** Institutions run training, embeddings, scoring and federation client inside their boundary; only encrypted updates leave the institution.
- **Hybrid:** Institutions perform local inference and embedding storage; federation coordination and model registry are hosted by the consortium.
- **Managed:** Smaller institutions use a managed orchestration plane while keeping raw data local; only updates cross boundaries. This lowers the barrier to entry for community banks and FinTechs.

6.6 Model Customization Policy

To maintain interoperability, the backbone architecture family, embedding dimension, update protocol, clipping/DP parameters and model contract are **fixed** across the federation. Institutions can customize only the local scoring head, thresholds and additional local features used solely in the head. This allows banks to tune risk appetite and integrate the model into their existing decision engines without breaking the federation.

7. Economics / Business Model

FGI's primary goal is to improve fraud detection accuracy and reduce false positives, thereby lowering losses and investigative costs for participating institutions. The consortium may operate on a membership fee and usage-based pricing model:

- **Membership or subscription fees:** Institutions pay for access to the shared platform, maintenance and governance.
- **API usage fees:** Charges based on the volume of online scoring requests or federated training rounds.
- **Data services:** Optional aggregated insights and risk signals (e.g., high-risk counterparties) may be offered through anonymized data products, subject to privacy and regulatory constraints.
- **Incentives:** Institutions that contribute to high quality updates receive discounts or elevated trust scores, increasing their influence in aggregation.

The model is scalable: costs scale with the number of clients and model complexity. Techniques such as quantization, compression and sparse updates reduce communication overhead; float16 embeddings enable tens of millions of nodes with manageable storage.

8. Risk & Mitigation

8.1 Privacy Risks

Federated learning reduces direct data sharing but still risks **membership inference** or **model inversion** attacks. Client-level differential privacy protects against inferring whether a particular institution participated in training[13]. Norm clipping limits each client's influence; secure aggregation ensures the server learns only aggregated sums[5]. Participants should also implement strong endpoint security and use hardware-backed enclaves to protect local training code.

8.2 Robustness Risks and Attacks

Adversaries could submit **poisoned updates** to backdoor the model or cause denial-of-service. Robust aggregation methods (median, trimmed mean, Krum, geometric median) provide tolerance to a fraction of corrupted clients[9]. Trust weighting reduces the impact of clients with anomalous update histories. Sentinel tests detect backdoors by evaluating the model on synthetic patterns (fan-in, fan-out, layering chains, cycles, smurfing); failing rounds are rolled back.

8.3 Non-IID and Heterogeneity

Performance degradation under non-IID splits is a known issue in federated GNNs[10]. To mitigate this, FGI mandates a canonical feature schema and windowing specification; cross-institution feature definitions are versioned and validated. Robust aggregation and

trust weighting help handle distributional heterogeneity, but long-term research may explore **client-adaptive aggregation** or **personalized federated GNNs**.

8.4 Model Risk Management and Governance

Regulators expect institutions to manage model risk. The Federal Reserve's SR 11-7 guidance emphasizes model validation, documentation and ongoing monitoring. FGI's compliance harness aligns with these expectations by enforcing deterministic serialization, hash verification, sentinel testing and audit logging. The NIST AI RMF's trustworthiness characteristics—valid & reliable, safe, secure, accountable, transparent, explainable, privacy-enhanced and fair[7]—should be considered throughout the lifecycle. The NIST CSF's functions (Govern, Identify, Protect, Detect, Respond, Recover) provide a high-level structure for managing cybersecurity risks[8].

9. Compliance & Governance

FGI supports regulators' expectations by providing:

- **SAR enablement:** The model supports investigators by identifying suspicious patterns and explaining contributing factors; this improves SAR narratives and ensures that suspicious activity reporting remains the cornerstone of AML programs[2].
- **Policy alignment:** The design aligns with FinCEN's AML/CFT priorities by targeting fraud and cybercrime threats[1]. It complements information sharing under Section 314(b) by providing risk signals across institutions without revealing customer data.
- **Trustworthy AI principles:** The system implements measures consistent with the NIST AI RMF characteristics (valid & reliable, safe, secure & resilient, accountable & transparent, explainable & interpretable, privacy-enhanced, fair)[7]. Explainability is delivered through pattern-level explanations (e.g., the contribution of neighbor bursts or layering chains) and a transparent model contract. Privacy is enhanced through secure aggregation and differential privacy.
- **Cybersecurity integration:** Governance aligns with NIST CSF 2.0 functions[8]: the consortium establishes policies and roles (Govern), institutions identify assets and data flows (Identify), protect data and models (Protect), detect anomalies via sentinel tests (Detect), respond to incidents through rollback and quarantine (Respond), and recover by restoring last-known-good models (Recover).

- **Model risk management:** Models are versioned, signed, validated and monitored. Institutions maintain documentation and justification for model usage; independent validation occurs within and across institutions; evaluation metrics (AUC, precision-recall, false-positive rates, typology recall, drift tests) are tracked per round.

10. Roadmap

Phase 1 – Pilot and Baseline Implementation (2026):

- Establish the consortium, finalize governance, and onboard early financial institutions.
- Implement the simplified person-to-person graph schema, baseline feature set and 2-layer temporal encoder with 128-dimensional embeddings.
- Integrate secure aggregation and robust aggregation primitives; deploy a sentinel test suite and compliance harness.
- Conduct pilot rounds focusing on known fraud typologies; collect metrics on improvement over local models.

Phase 2 – Expansion and Heterogeneous Graphs (2027):

- Onboard additional institutions, including payment processors and FinTechs.
- Extend the graph schema to include accounts, devices and merchants; adopt heterogeneous GNN architectures (R-GCN/HGT) with relation-specific transformations[12].
- Introduce personalized federated GNNs or client-adaptive aggregation to handle distributional heterogeneity.
- Explore advanced self-supervised objectives and contrastive learning for improved robustness.

Phase 3 – Ecosystem Integration and Continuous Learning (2028 and beyond):

- Integrate FGI scores into payment rails, core banking systems and real-time payment networks.
- Develop APIs for regulators and law-enforcement partners to receive aggregated risk signals, subject to legal constraints.
- Automate model lifecycle management: continuous training, evaluation, rollback and update distribution.

- Expand to other financial crime domains (e.g., trade-based money laundering, sanctions evasion) and incorporate external data sources (adverse media, crypto transactions) while preserving privacy.

Conclusion

Financial fraud and illicit finance are dynamic, networked problems that no single institution can fully observe. **Federated Graph Intelligence** provides a privacy-preserving framework for institutions to learn from each other without sharing customer data. By combining graph neural networks, temporal message passing, robust and secure federated learning, and a strong compliance harness, FGI aims to improve detection rates, reduce false positives and support regulatory obligations. The design aligns with AML/CFT priorities[1], trusts the SAR system[2], and adheres to the principles of trustworthy AI and cybersecurity risk management[7][8]. With careful implementation and governance, FGI can become a cornerstone of collaborative fraud detection for the financial services ecosystem.

References

(AIRC), N. A. (n.d.). *AI Risks and Trustworthiness*. NIST AI Resource Center (AIRC). From https://airc.nist.gov/airmf-resources/airmf/3-sec-characteristics/

Bonawitz, K. (n.d.). Practical Secure Aggregation for Federated Learning on User-Held Data. *arXiv*. doi:1611.04482

ffiec.gov. (n.d.). *FFIEC BSA/AML Assessing Compliance with BSA Regulatory Requirements - Suspicious Activity Reporting*. From https://bsaaml.ffiec.gov/manual/AssessingComplianceWithBSARegulatoryRequirements/04

FinCEN.gov. (n.d.). *FinCEN Issues First National AML/CFT Priorities and Accompanying Statements*. FinCEN.gov. From https://www.fincen.gov/news/news-releases/fincen-issues-first-national-amlcft-priorities-and-accompanying-statements

Geyer, R. C. (2018). Differentially Private Federated Learning: AClient Level Perspective. doi:1712.07557

Graph Neural Networks for Financial Fraud Detection: A Review. (2024). *Higher Education Press*. doi:10.1007/s11704-024-40474-y

Hamilton, W. L. (2018). Inductive Representation Learning on Large Graphs. doi:1706.02216

He, C. (2021). FedGraphNN: A Federated Learning Benchmark System for Graph Neural Networks. doi:2104.07145

McMahan, B. (n.d.). Communication-Efficient Learning of Deep Networks from Decentralized Data. *arxviz*. doi:1602.05629

NIST. (n.d.). The NIST Cybersecurity Framework (CSF) 2.0. doi:doi.org/10.6028/NIST.CSWP.29