

DP-RTFL: Differentially Private Resilient Temporal Federated Learning for Trustworthy AI in Regulated Industries

Abhijit Talluri

RTFL Project Contributor

Email: talluri.abhijit@gmail.com

Developmental insights reflect work up to March 1, 2024.

Abstract—Federated Learning (FL) has emerged as a critical paradigm for enabling privacy-preserving machine learning, particularly in regulated sectors such as finance and healthcare. However, standard FL strategies often encounter significant operational challenges related to fault tolerance, system resilience against concurrent client and server failures, and the provision of robust, verifiable privacy guarantees essential for handling sensitive data. These deficiencies can lead to training disruptions, data loss, compromised model integrity, and non-compliance with data protection regulations (e.g., GDPR, CCPA). This paper introduces Differentially Private Resilient Temporal Federated Learning (DP-RTFL), an advanced FL framework designed to ensure training continuity, precise state recovery, and strong data privacy. DP-RTFL integrates local Differential Privacy (LDP) at the client level with resilient temporal state management and integrity verification mechanisms, such as hash-based commitments (referred to as Zero-Knowledge Integrity Proofs or ZKIPs in this context). The framework is particularly suited for critical applications like credit risk assessment using sensitive financial data, aiming to be operationally robust, auditable, and scalable for enterprise AI deployments. The implementation of the DP-RTFL framework is available as open-source [1].

Index Terms—Federated Learning, Resilient Systems, Differential Privacy, Temporal State Management, Integrity Verification, Information Theory, Regulatory Compliance, Fault Tolerance, Credit Risk Assessment, Secure AI, Trustworthy AI, Enterprise AI, Banking Technology.

I. INTRODUCTION

Federated Learning (FL) enables collaborative AI model development across distributed data sources without centralizing sensitive raw data [2]. This paradigm is particularly promising for industries like finance and healthcare, which handle highly sensitive information and are subject to strict data privacy regulations. Despite its potential, the widespread adoption of FL in mission-critical sectors is hindered by significant challenges, primarily operational resilience against concurrent client and server failures, and the lack of robust, verifiable privacy guarantees. Standard FL systems may suffer training disruptions, data loss, and model integrity issues, and often struggle to comply with data protection mandates such as GDPR [3] and CCPA.

To address these deficiencies, we propose Differentially Private Resilient Temporal Federated Learning (DP-RTFL). DP-RTFL is a comprehensive framework designed for continuous, auditable, and privacy-preserving training on sensitive

datasets, such as financial records for credit risk assessment (e.g., the Credit Card Approval Prediction dataset from Kaggle [4]). Our approach enhances the original Resilient Temporal Federated Learning (RTFL) concept by deeply integrating Local Differential Privacy (LDP) at the client level. This ensures that individual contributions to model updates are cryptographically protected before aggregation, complementing the framework’s inherent resilience capabilities. The software implementation and experimental code for DP-RTFL are publicly available [1].

The key contributions of DP-RTFL include:

- **Local Differential Privacy (LDP) Integration:** Client-side application of (ϵ, δ) -Differential Privacy to model updates, formally limiting information leakage about individual data records and supporting regulatory compliance.
- **Temporal Checkpoint Manifold (TCM):** A distributed, chronologically ordered log of global model states and client contributions, allowing precise rollback for recovery and audibility.
- **Differential State Synchronization (DSS):** Clients transmit only model parameter changes (deltas), reducing communication overhead, which is especially crucial when updates are expanded by DP noise.
- **Adaptive Role Reassignment Protocol (ARRP):** A dynamic protocol enabling eligible clients to assume coordination duties if the central server fails, ensuring training continuity.
- **Zero-Knowledge Integrity Proofs (ZKIP):** A mechanism (in this implementation, a hash-based commitment) accompanying model updates, allowing verification of integrity and origin without revealing sensitive data, thereby enhancing verifiability.
- **Entropy-Based Corruption Detection (EBCD):** Utilizes information-theoretic principles (statistical moments) to identify corrupted or anomalous model updates, even in the presence of DP noise.

This paper details the DP-RTFL framework, its components, system architecture, application to credit risk assessment, and a comprehensive evaluation strategy.

II. RELATED WORK

Federated Learning, introduced by McMahan et al. [2], laid the groundwork for distributed machine learning with data decentralization. Kairouz et al. [5] provide an extensive overview of advances and open problems in FL, highlighting challenges in efficiency, robustness, and privacy.

The need for privacy in FL has led to the integration of Differential Privacy (DP) [6], [7]. Local Differential Privacy (LDP) is particularly relevant as it provides privacy guarantees at the client level before data (or model updates) leave the user’s device. Works such as Truex et al. [8] and Sun et al. [9] have explored LDP in FL, though practical implementations face challenges with utility and high dimensionality. DP-RTFL aims to provide a practical LDP integration by carefully calibrating noise and combining it with other protective measures.

Resilience and fault tolerance are critical for enterprise FL deployments. While some works focus on Byzantine fault tolerance in aggregation [10], [11], DP-RTFL’s ARRP addresses server/coordinator failures, and its TCM provides a general mechanism for state recovery from various faults. The concept of temporal checkpointing for resilience is also explored in distributed systems, although its specific application as a manifold in FL, as in TCM, is a nuanced contribution.

Integrity verification in FL is crucial. While full Zero-Knowledge Proofs (ZKPs) can offer strong guarantees for verifiable computation in machine learning (ZKML) [12], [13], they can be computationally intensive. DP-RTFL employs a lighter-weight hash-based commitment scheme as its ZKIP component (`zkip.py`), akin to those used for data integrity checks [14], to verify update authenticity and integrity from known participants in a federated setup. Naseri et al. [15] discuss the broader need for robust and verifiable FL.

Anomaly detection using statistical properties, such as EBCD’s use of variance, kurtosis, and skewness (`ebcd.py`), draws from established statistical process control and data mining techniques [16], [17]. Adapting these for noisy, high-dimensional model parameter distributions in DP-FL is a specific challenge that EBCD addresses.

DP-RTFL distinguishes itself by holistically combining these elements—LDP, temporal resilience, adaptive coordination, hash-based integrity proofs, and information-theoretic anomaly detection—into a unified framework tailored for regulated industries.

III. THE DP-RTFL FRAMEWORK

DP-RTFL is designed with several key components working in concert to provide a robust, private, and auditable FL environment. These modules, illustrated conceptually in the overall system architecture (see Figure 1), interact to deliver the framework’s capabilities.

A. Local Differential Privacy (LDP)

LDP is applied client-side to protect individual data contributions before model updates are sent to the aggregator.

- **Mechanism:** Each client, after local training with libraries like scikit-learn for model implementation and NumPy for

numerical operations (`fl_client.py`), computes model parameter deltas. These deltas are then privatized using an (ϵ, δ) -Differential Privacy mechanism. This involves L2 norm clipping of the deltas, followed by the addition of Gaussian noise calibrated to the sensitivity of the clipped deltas and the chosen privacy budget (ϵ, δ) .

- **Purpose:** This ensures that the server or any other entity cannot infer significant information about any individual’s data from their (noisy) model update, which is crucial for regulatory compliance and user trust.
- **Configuration:** The privacy budget (ϵ, δ) and clipping norm are configurable per client or per round, allowing a trade-off between privacy and model utility.

Algorithm 1 outlines the client-side LDP process, as implemented in `fl_client.py`.

Algorithm 1 Client-Side LDP for Model Deltas

- 1: **Input:** Local model parameters θ_{local} , base model parameters θ_{base} , L2 clip bound C , privacy budget (ϵ, δ)
 - 2: Compute delta: $\Delta \leftarrow \theta_{local} - \theta_{base}$ (via DSS)
 - 3: Compute L2 norm: $N_2(\Delta) \leftarrow \sqrt{\sum \Delta_i^2}$
 - 4: Clip delta: $\Delta_{clipped} \leftarrow \Delta \cdot \min(1, C / (N_2(\Delta) + 1e-6))$
 - 5: Calculate noise stddev: $\sigma \leftarrow \frac{C\sqrt{2\ln(1.25/\delta)}}{\epsilon}$ (if $\epsilon > 0$)
 - 6: Generate Gaussian noise: $Noise \sim \mathcal{N}(0, \sigma^2)$
 - 7: Add noise: $\Delta_{private} \leftarrow \Delta_{clipped} + Noise$
 - 8: **Output:** Private delta $\Delta_{private}$
-

B. Temporal Checkpoint Manifold (TCM)

The TCM (`tcmm.py`) provides robust, auditable, and precise recovery of global model states and training history. It maintains a chronological log of all global model states, client update summaries (including DP parameters used), and coordinator actions. Each entry is timestamped and hashed for integrity, enabling precise rollback and providing comprehensive audit trails.

C. Differential State Synchronization (DSS)

DSS (`dss.py`) aims to reduce communication overhead by transmitting only model parameter deltas. Clients compute the difference (delta) between their updated local model and the global model received. Only this delta is transmitted, minimizing data transfer, especially when updates are expanded by DP noise.

D. Adaptive Role Reassignment Protocol (ARRP)

ARRP (`arrp.py`) ensures uninterrupted training by dynamically reassigning the coordinator role if the central server fails. It involves automatic detection of server failure and an election process among eligible, active clients to select a new coordinator, thereby providing seamless failover.

E. Zero-Knowledge Integrity Proofs (ZKIP)

The ZKIP component (`zkip.py`) in DP-RTFL uses a hash-based commitment scheme. Each client generates a SHA256 hash of its serialized (noisy) model delta concatenated with a

shared secret. This proof accompanies the delta, allowing the coordinator to verify its integrity and origin from a legitimate participant without needing to know the original (pre-noise) delta. This process ensures that updates are authentic and untampered.

F. Entropy-Based Corruption Detection (EBCD)

EBCD (`ebcd.py`) detects potentially corrupted model updates by monitoring statistical moments (variance, kurtosis, and skewness) of model parameters. It establishes a dynamic baseline from initial client models and flags deviations that exceed a tolerance factor, helping identify anomalies beyond expected DP noise. The use of such moments for anomaly detection is a common technique in statistical analysis [17], [16].

G. Early Stopping

Standard early stopping (`earlystop.py`) is employed at client and server levels to prevent overfitting and improve efficiency. It halts training or restores the best model if validation performance (loss for clients, accuracy for server) ceases to improve for a configurable patience period.

IV. THREAT MODEL AND ASSUMPTIONS

DP-RTFL operates under the following threat model:

- **Clients:** Assumed to be honest-but-curious regarding their own data. However, a fraction of clients could be malicious (Byzantine), attempting to degrade model performance or integrity. LDP provides protection against inference from their updates regardless of server behavior. EBCD and ZKIPs aim to mitigate impacts from overtly malicious updates.
- **Server/Coordinator:** The central server or an ARRP-elected coordinator is assumed to be honest-but-curious. It will follow the protocol but might attempt to infer information about individual client data from received updates. LDP is the primary defense against this. The server is also a point of failure, addressed by ARRP and TCM.
- **External Adversaries:** May attempt to eavesdrop on communication channels or compromise clients/server. Secure communication channels (e.g., TLS, not explicitly part of this framework’s core logic but assumed in a real deployment) would be necessary. ZKIPs help ensure update integrity against tampering in transit.

It is assumed that cryptographic primitives (hashing for ZKIP) are secure and that the shared secret for ZKIP is managed securely among legitimate participants.

V. SYSTEM ARCHITECTURE

The DP-RTFL system architecture is designed for modularity and resilience. A conceptual overview, illustrating the interaction between data sources, core FL processes, advanced protocol modules, and reporting utilities, is presented in Figure 1. This diagram visually represents the system components detailed in the project’s `README.md` file.

The main components are:

- 1) **Data Sources:** Distributed datasets, e.g., `application_record.csv` and `credit_record.csv` for credit risk from a source like Kaggle [4].
- 2) **Application Core (`main.py`):** Orchestrates the simulation, client/server instances, metrics collection, and integrates all DP-RTFL components.
- 3) **Federated Learning Process Components:**
 - *FL Server Logic (`fl_server.py`):* Manages aggregation, ZKIP verification, server-side EBCD, TCM, ARRP, and global evaluation.
 - *FL Client Logic (`fl_client.py`):* Handles local training, LDP, ZKIP generation, and DSS interaction.
 - *Data Utilities (`data_utils.py`):* Data loading, preprocessing, and client splitting.
- 4) **Advanced FL Protocol Modules:** TCM, DSS, ARRP, ZKIP, EBCD, and Early Stopping, each implemented in their respective Python files (e.g., `tcn.py`, `dss.py`).
- 5) **Reporting (`charting.py`):** Generates plots for metrics visualization.

VI. METHODOLOGY AND EXPERIMENTAL SETUP

The DP-RTFL methodology encompasses data handling, the federated training process, and simulation parameters.

A. Data Preparation and Distribution

The primary dataset used is the “Credit Card Approval Prediction” dataset from Kaggle [4], which includes `application_record.csv` and `credit_record.csv`. Preprocessing steps involve merging these files; deriving a binary target variable for credit risk (good/bad); handling missing values (median imputation for numerical, mode for categorical); and transforming features (e.g., `DAYS_BIRTH` to years, `DAYS_EMPLOYED` to years of employment, and normalizing negative day counts). Categorical features are one-hot encoded, and numerical features are standardized using scikit-learn’s `ColumnTransformer` and `StandardScaler`. The processed data is split into a global training set and a test set (80/20 split). The global training set is then distributed (potentially non-IID) among `NUM_CLIENTS`. Clients may further split their local data for validation.

B. Federated Training and Simulation

The simulation (`main.py`) runs for `NUM_ROUNDS`. In each round:

- 1) The server (or ARRP coordinator) provides global parameters. TCM is used for recovery if the server fails to provide parameters.
- 2) Active clients (simulating potential dropouts) train a local `SGDClassifier` model (from scikit-learn) for `CLIENT_EPOCHS`.
- 3) Deltas are computed (DSS), privatized (LDP with `DP_EPSILON`, `DP_DELTA`, `DP_L2_NORM_CLIP`), and a ZKIP is generated.

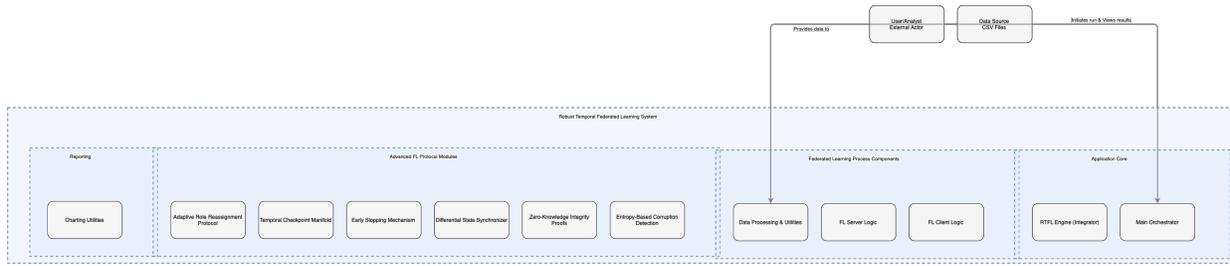


Fig. 1: DP-RTFL Conceptual System Diagram. This diagram visually represents the architecture detailed in the project’s README.md, showing data flow and component interactions (adapted from the project’s README.md).

- 4) The server verifies ZKIPs, aggregates valid deltas (weighted by client data sizes), and updates the global model.
 - 5) EBCD checks the global model, and TCM logs the state.
 - 6) Metrics (accuracy, F1-score, AUC, etc.) are collected.
- Key parameters, such as `NUM_CLIENTS=5`, `NUM_ROUNDS=10`, `CLIENT_EPOCHS=3`, and DP parameters, are specified in `main.py`.

VII. EVALUATION METRICS AND EXPECTED RESULTS

The DP-RTFL framework is evaluated on several axes.

A. Privacy-Utility Trade-off

This is assessed by model performance (Accuracy, F1-score, AUC-ROC on the test set) against varying DP budgets (ϵ , δ) and L2 clipping norms.

- **Expected Outcome:** Figure 2 would show model convergence. Higher privacy (lower ϵ) is expected to result in a graceful degradation of utility. Figure 3, visualizing mean DP noise standard deviation per round, should show higher noise for stricter privacy budgets.

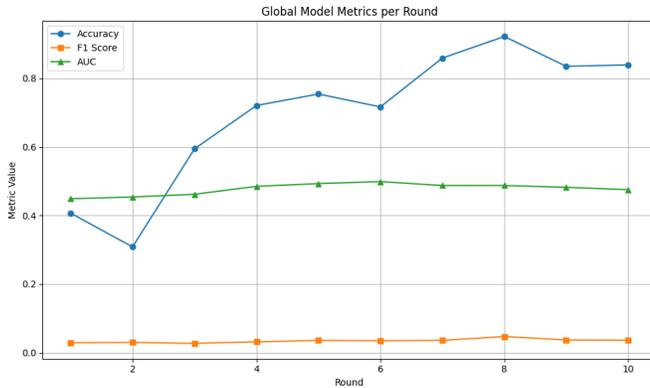


Fig. 2: Global Model Metrics (Accuracy, F1, AUC) per Round. Expected to show convergence and impact of DP budget on final performance.

B. Resilience and Recovery Precision

Training continuity under simulated failures and TCM recovery fidelity are key metrics.

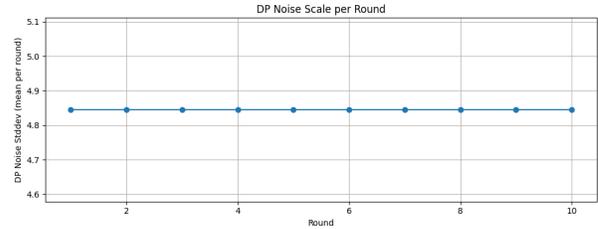


Fig. 3: DP Noise Scale (Mean Stddev) per Round. Expected to correlate with chosen privacy parameters (ϵ , δ).

- **Expected Outcome:** Figure 4 (plotting server status and coordinator ID) should demonstrate ARR’s ability to maintain a coordinator. Figure 5 (TCM states per round) should show consistent logging. Successful TCM recovery from a mid-training round, as simulated in `main.py`, would validate recovery precision.

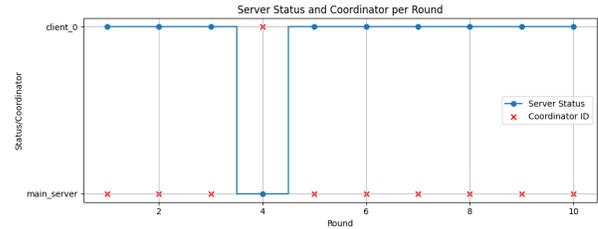


Fig. 4: Server Status and Coordinator ID per Round. Expected to show ARR’s dynamic role reassignment during simulated failures.

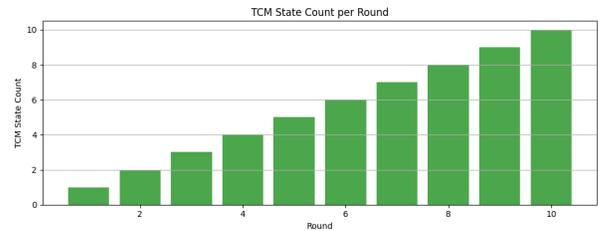


Fig. 5: TCM State Count per Round. Expected to show steady accumulation of states, vital for audit and recovery.

C. Communication, Integrity, and Anomaly Detection

The overhead of DSS, ZKIP, and EBCD effectiveness are measured.

- **Expected Outcome:** Figure 6 (L2 norm of aggregated deltas) would show update magnitudes. Figure 7 (ZKIP failures per round) should be minimal in normal operation. Figure 8 (variance, kurtosis, and skewness of global weights) and Fig. 9 (EBCD alerts) should indicate stability and detection of simulated anomalies.

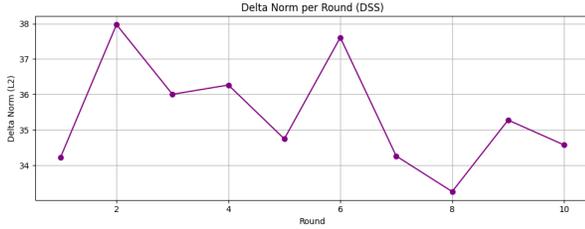


Fig. 6: Aggregated Delta Norm (L2) per Round. Reflects the magnitude of global model updates via DSS.

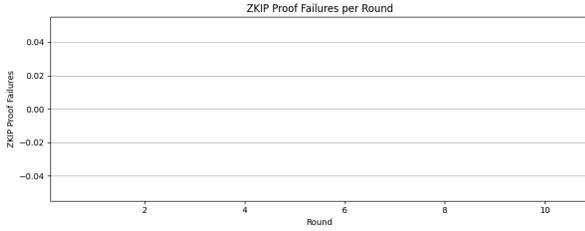


Fig. 7: ZKIP Proof Failures per Round. Expected to be low, indicating update integrity.

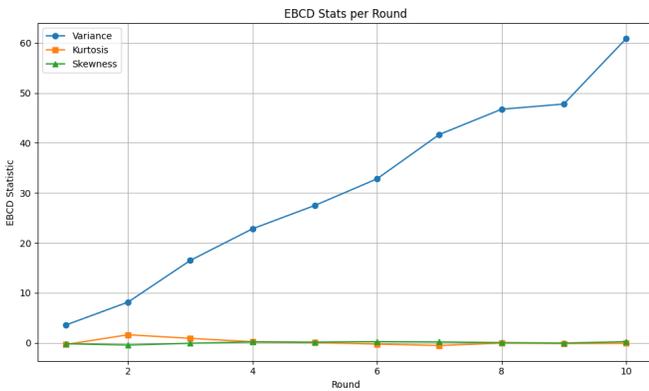


Fig. 8: EBCD Statistics (Variance, Kurtosis, Skewness) per Round. Monitors global model parameter distribution.

D. Training Stability and Auditability

Early stopping effectiveness and TCM’s audit trail completeness are assessed.

- **Expected Outcome:** Figure 10 (best validation accuracy from server’s early stopping) should demonstrate prevention

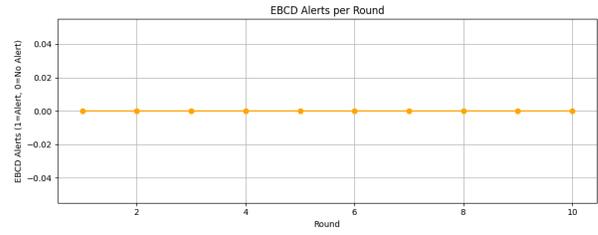


Fig. 9: EBCD Alerts per Round. Indicates rounds where potential anomalies were flagged.

of overfitting. The TCM logs themselves (content not plotted but functionality tested by recovery) provide the basis for auditability.

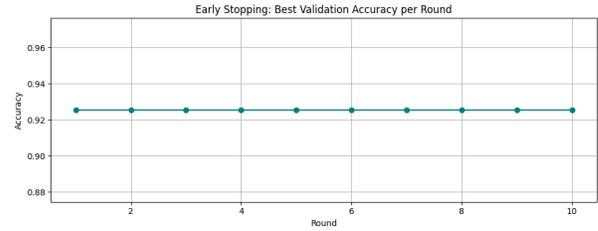


Fig. 10: Early Stopping: Best Server Validation Accuracy per Round. Shows effectiveness in finding optimal training duration.

Per-client plots (e.g., update norms, EBCD stats, ZKIP status from `charting.py`) would offer granular insights into individual client contributions and adherence to protocols.

VIII. LIMITATIONS

While DP-RTFL offers a comprehensive approach, certain limitations exist:

- The ZKIP mechanism implemented is a hash-based commitment, which relies on a shared secret and primarily ensures integrity and authenticity from known participants, rather than providing full zero-knowledge computational proofs against arbitrary verifiers.
- The framework’s evaluation is currently simulation-based. Real-world network conditions, diverse hardware, and true adversarial behaviors might present additional challenges.
- The scalability of some components (e.g., TCM logging, EBCD baseline establishment with many clients) needs further testing in very large-scale federations.
- The choice of DP parameters (ϵ , δ , clipping bound) involves a trade-off that might require careful tuning for specific applications and datasets to achieve optimal privacy and utility.

IX. ETHICAL CONSIDERATIONS

The development and deployment of FL systems, especially in sensitive areas like finance, carry significant ethical responsibilities.

- **Privacy Preservation:** LDP is a core component aiming to protect individual financial data. However, the choice of privacy parameters and the potential for re-identification in high-dimensional sparse data, even with DP, must be continually assessed.
- **Fairness and Bias:** FL models can inherit or even exacerbate biases present in distributed client data. While not a direct focus of DP-RTFL's current components, ensuring fairness and mitigating bias in credit risk assessment models is a critical ongoing concern that necessitates additional mechanisms.
- **Transparency and Auditability:** TCM aims to provide audit trails, contributing to transparency. Nevertheless, the complexity of FL and DP can make full transparency to end-users challenging.
- **Security:** While resilience and integrity are addressed, sophisticated attacks against FL systems represent an evolving research area.

Continuous vigilance and adherence to ethical AI principles are necessary throughout the lifecycle of such systems.

X. CONCLUSION

DP-RTFL presents a holistic framework for conducting Federated Learning in environments with stringent requirements for privacy, resilience, and verifiability. By integrating Local Differential Privacy, Temporal Checkpoint Manifolds, Differential State Synchronization, Adaptive Role Reassignment, hash-based integrity proofs (ZKIP), and Entropy-Based Corruption Detection, DP-RTFL addresses critical operational challenges that have historically hindered FL adoption in sensitive domains like finance and healthcare.

The framework's design emphasizes training continuity, precise state recovery, and formal data privacy guarantees, rendering it suitable for applications such as credit risk assessment using sensitive financial data. The potential scientific impact lies in advancing trustworthy distributed AI. Practically, DP-RTFL aims to enable FL adoption for high-stakes applications by satisfying regulatory compliance and reducing operational risks.

Future work will focus on extending LDP schemes and exploring advanced privacy accounting. Investigation into more sophisticated, yet practical, ZKP constructions is warranted. Extensive testing in enterprise-grade environments with diverse datasets is crucial. Exploring fairness-aware mechanisms within the DP-RTFL framework and adapting components for emerging hardware, such as confidential computing enclaves, are other promising directions. Addressing the identified limitations, particularly concerning real-world deployment complexities and advanced adversarial scenarios, will be key to maturing the framework.

REFERENCES

[1] A. Talluri, "DP-RTFL: Differentially Private Resilient Temporal Federated Learning Framework," <https://github.com/abhtall/federated-credit-risk-rtfl.git>, mar 2024, accessed: May 25, 2025.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*. PMLR, 2017, pp. 1273–1282.

[3] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, 2016.

[4] Rikesh Bhattacharyya and Others, "Credit Card Approval Prediction," 2020, accessed: May 24, 2025. [Online]. Available: <https://www.kaggle.com/datasets/rikdifos/credit-card-approval-prediction>

[5] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," *Theory of cryptography conference*, pp. 265–284, 2006.

[7] C. Dwork and A. Roth, *The algorithmic foundations of differential privacy*. Now Publishers Inc., 2014.

[8] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Hybrid alpha-beta-gamma: A differentially private federated learning framework," in *Proceedings of the 2019 ACM International Workshop on Security in Machine Learning*, 2019, pp. 1–11.

[9] L. Sun, L. Lyu, and T. Li, "Ldp-fl: Practical private aggregation in federated learning with local differential privacy," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 11, pp. 9797–9805, 2021.

[10] P. Blanchard, E. M. E. Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in neural information processing systems*, vol. 30, 2017.

[11] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*. PMLR, 2018, pp. 5650–5659.

[12] Z. Ghodsi, R. Raskar, and M. Dror, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," *arXiv preprint arXiv:1702.02221*, 2017.

[13] Y. Liu, H. Zhu, S. Liu, Y. Zhao, R. Lu, and X. Lin, "Jama (jama): Verifiable function secret sharing for privacy-preserving federated learning," in *2021 IEEE International Conference on Communications (ICC)*. IEEE, 2021, pp. 1–6.

[14] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology — CRYPTO '89 Proceedings*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Springer Berlin Heidelberg, 1990, pp. 369–378.

[15] M. Naseri, J. Hayes, and E. De Cristofaro, "Towards the robustness of differentially private federated learning," *arXiv preprint arXiv:2301.09369*, 2023.

[16] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.

[17] J. D. Jobson, *Applied multivariate data analysis: Volume II: Categorical and multivariate methods*. Springer Science & Business Media, 1992, vol. 2.