International Journal of Web of Multidisciplinary Studies
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
website: https://ijwos.com
Volume 3 Issue 8 , August 2026
E-ISSN : 3049-2424
DOI: https://doi.org/10.71366/ijwos

SecureLedger: A Hybrid Blockchain–Machine Learning Framework for Real-Time Fraud Detection and Tamper-Evident Financial Transaction Recording

Krupa Rani S[1], D K Soumya[2], Heeba A[3], Chandana N S[4]
[1] *Assistant Professor , Faculty of Computing & IT, GM University, Davanagere, Karnataka, India*
[2,3,4] *2nd year MCA Student, Faculty of Computing & IT, GM University, Davanagere, Karnataka, India.*

Article Info

Article History:

Published: 12 August 2026

Publication Issue:
Volume 3, Issue 8
August-2026

Page Number:
199-212

Corresponding Author:
Krupa Rani S

Abstract:

There is a rise in losses resulting from transaction fraud in financial institutions; Decentralized Finance (Defi) platforms have suffered about $10.1 billion worth of loss during 31 major attacks from 2020 through 2023 [3]. In this paper, SecureLedger is introduced, which is a working Flask framework integrating a Random Forest classifier for fraud prediction together with a custom-designed SHA-256 chained hash ledger such that each transaction is fraud scored and permanently written to ledger as a block. With respect to a synthetic dataset of 2,000 transactions, the model achieved 99.67%, 98.61%, and 0.9924 in terms of accuracy, precision/recall/F1-score, and Area Under the Curve (AUC), respectively, where account age and transaction velocity were found to be the main predictors. The live admin dashboard and chain integrity check functionality in the running prototype are validated to show the true state of the ledger as well as detecting any intentional tampering. Compared to other six recent works in the application of blockchain technology in financial security, SecureLedger gives up architecture complexity for clarity and reproducibility.

Keywords: Blockchain, Financial Security, Fraud Detection, Random Forest, SHA-256, Hash Chain, Machine Learning, Proof-of-Work

1. INTRODUCTION

The digital transformation of financial services has increased the potential areas of fraud in spite of the increasing implementation of distributed ledger technology in capital formation, trading in securities, financial analysis, and investment management due to their immutable nature [3], which is supported by the findings of bibliometric studies highlighting the fast-growing amount of research output on blockchain-based finance [7]. The losses resulting from 31 prominent security incidents in the decentralized finance sector reached about $10.1 billion during the period from 2020 to 2023 [3], proving that the blockchain implementation in itself does not reduce the threat of fraud. Machine learning methods have become the primary means of fraud detection: capsule network synthetic dataset generation [1], privacy-preserving incremental classification [2], lightweight deep learning [4], transformer-CNN hybrid [5], and graph convolutional neural networks [6] have already shown more than 97% accuracy but have a complicated architecture, require large proprietary datasets, and are unreplicable as a unified system.

There have been very few examples where the live decision made by the classifier is integrated within the tamper-proof block of one single system. Blockchain implementation has always been used either for storing the metadata or incentives [1], [2] or at platform level [3], [4], [5]. In this paper, we aim at filling this void with SecureLedger, a Flask-based web application where each and every transaction is scored by the Random Forest classifier and immediately embedded into the SHA-256 hash-linked ledger in the form of a block. The contributions of this paper are:The rest of this paper is structured as follows. Section II presents previous work. Section III presents the SecureLedger architecture. Section IV formulates the algorithms used by SecureLedger mathematically. Section V describes the

implementation process. Section VI presents the experimental results obtained directly from the system. Section VII presents a discussion on the results and the limitations of the prototype.

2. RELATED WORK

A. Blockchain-Enabled Machine Learning Pipelines

Wang et al. [1] developed a blockchain architecture based on Capsule Networks for the generation of synthetic samples of the minority class in credit card fraud detection (284,807 transactions, 0.17% fraud) in a small memory-efficient ASTORE smart-contract representation of the model; synthetic samples have been shown to enhance the trade-off between precision and recall compared to the baseline approaches, while the authors highlight the problem of memory limits during edge deployment. Pranto et al. [2] designed a privacy-preserving architecture with an incremental Passive-Aggressive Classifier with SMOTE oversampling and a difficulty adaptive smart-contract incentive mechanism, which was able to reach 93.64-96.86% accuracy on the PaySim dataset of 6.3 million transactions, noting that the proof-of-work mining time increases significantly as the difficulty of mining is increased (less than 50 ms with difficulty 2 against ≈70 s with difficulty 5). These pipeline level results were confirmed by additional surveys, not involving blockchain technology: Alarfaj et al. [8] benchmarked ensemble and deep learning classifiers on popular fraud datasets, while Motie and Raahemi [9] reviewed graph-based fraud detection approaches.

B. Deep Learning and Transformer-Based Approaches

Combining MobileNet-based CNN architecture with LSTM layers and the Hyperledger Fabric blockchain ledger, Darwish et al. [4] reach 93.2–97.4% accuracy with a mere 80 MB memory consumption yet suffering from 14.3% loss in accuracy during adversarial attacks. Fnu et al. [5] integrate transformer-based feature extraction (RoBERTa), metaheuristic-based feature selection (EAFO), and a Spatial CNN architecture together with Ethereum/Hyperledger integration, obtaining the highest accuracy among all solutions in this paper (99.69–99.77%) at the price of the complexity of quadratic order of their transformer operations. However, tree-based ensemble learning architectures prove to be powerful and inexpensive alternatives to such deep architectures: Xu et al.

[10] propose an efficient boosting-based fraud detection method consuming fewer computational resources compared to similar neural networks, whereas Lo et al. [11] propose self-supervised graph-based blockchain embedding for detecting money laundering in Bitcoin blockchains, showing more direct coupling of blockchain and ML methods than [1], [2].

C. Graph-Based Fraud Detection and Platform Surveys

Islam et al. [6] present a Layer-Weighted Graph Convolutional Network demonstrating 98% accuracy and good noise robustness on a 10 million transaction synthetic dataset, without any blockchain, but clearly pointing out the black-box nature of the model as a regulation hurdle. Wu et al. [3] survey twelve blockchain-based platforms used in four different financial application areas, highlighting the reduction in settlement time (from three days to ≈10 minutes), along with 31 cases of DeFi attacks amounting to losses of $10.1 billion, while mentioning that performance and privacy testing of the surveyed literature quantitatively is insufficient. Smart contract trust issues are also examined from a platform perspective: Kushwaha et al. [12] review categories of Ethereum vulnerabilities and their detection tools, and Guru et al. [13] conduct surveys about attacks targeting consensus protocols in PoW, PoS, and PBFT — which is pertinent to this work (III-C).

These six articles, together with SecureLedger, are shown in Table I. The complexity of the models and size of the dataset are associated with accuracy, but blockchain usage in the literature surveyed above is limited mostly to either storing metadata or conducting platform-level analysis, not to building a single-runnable system providing fraud decision with tamper-evident record

TABLE I. COMPARISON OF BLOCKCHAIN–ML FRAUD DETECTION METHODS

Ref.	**ML Model**	**Blockchain**	**Dataset**	**Accuracy**
[1]	Capsule Net + GBT	Smart Contract	Credit Card (285K)	PR Improved
[2]	Passive-Aggressive	PoW	PaySim (6.3M)	93.64–96.86 %
[3]	Survey	Multi-platform	Literature	N/A
[4]	Mobile Net + LSTM	Hyperledger	IEEE-CIS, PaySim	93.2–97.4%
[5]	RoBERTa + SD-CNN	Ethereum + Hyperledger	IEEE-CIS	99.69–99.77 %
[6]	LWG-GCN	None	SIFT (10M)	98.0%
Proposed	**Random Forest**	**SHA-256 + PoW**	**Synthetic (2K)**	**99.67 %**

3. PROPOSED SYSTEM

SECURE LEDGER ARCHITECTURE

SecureLedger is organized into four cooperating modules — a user module, an administrative module, a fraud-detection module, and a blockchain ledger module — orchestrated by a

Flask application layer, as shown in Fig. 1.

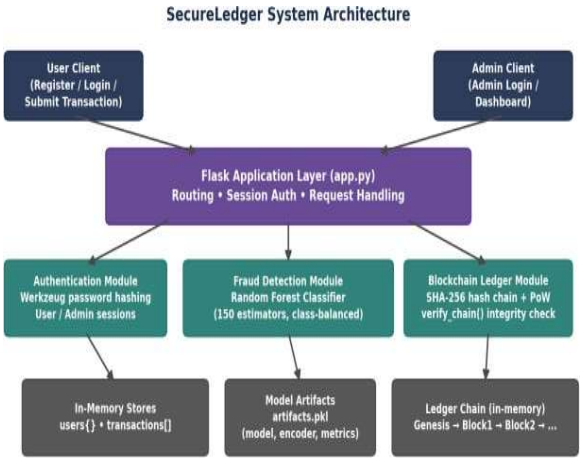

Fig. 1. SecureLedger system architecture.

A. User and Administrative Modules

Registration, logging-in through session and transaction submission form, which includes six inputs like amount, account age, velocity of transactions in 24 hours, time of day, location mismatch indicator and transaction types, is provided by the user module. Hashing of passwords is done using security features of Werkzeug. For administrative privileges, a different login and session flag is used by the administrative module.

B. Fraud Detection Module

The trained random forest classifier analyzes each transaction that is fed to it and generates a fraud probability. This probability is then transformed using a decision boundary of 0.5 to give a binary label. This label, along with the probability, is then appended to the transaction and fed to the blockchain module. The random forest classifier needs to be trained only once and stored on disk for future use.

C. Blockchain Ledger Module

Each recorded transaction gets added to a bespoke, scratch-built blockchain (Block and Blockchain classes). Every block contains its index, time stamp, data (transaction and prediction), hash of the previous block, a nonce, and its SHA-256 hash.

Proof-of-work mechanism incrementing the nonce until the resulting hash string starts with a certain number of zero hex digits (difficulty = 2) makes block generation rapid enough to be completed within one web request. verify_chain() function recalculates hashes of all blocks and validates previous hash linkage; any discrepancy will lead to revealing the exact block index where the manipulation occurs.

D. Security Mechanisms

SecureLedger uses three security mechanisms: hashing chain with proof-of-work for integrity, password hashing with salt for credentials management, and session-based access control with two different trust domains – users and administrators. In contrast to [1] and [2], where the blockchain serves metadata or incentives purposes, the ledger here is responsible for fraud decision, thus recording not only decision but its integrity as well.

4. MATHEMATICAL FORMULATION

This section outlines the implementation details of the algorithmic elements in SecureLedger: hash chain cryptography, proof-of-work, Random Forest classification, and Gini index splitting method for the scikit-learn default RandomForestClassifier, along with the classification measures provided in Section VI.

A. Hash Chain Cryptography and Proof-of-Work

Each block is chained to its previous block according to (1) and accepted into the blockchain chain when a nonce that meets the difficulty measure in (2) is computed.

$$H_i = \text{SHA-256}(I_i \| T_i \| D_i \| H_{i-1} \| N_i) \quad (1)$$

H_i, I_i, T_i, D_i, H_{i-1}, N_i — hash, index, timestamp, transaction-and-prediction data, hash of the previous block, and the nonce for block i; ‖ represents concatenation.

$$H_i < T, \qquad T = 2^{256-4k} \quad (2)$$

T, k — the acceptance threshold and the number of leading zero hexadecimal digits that must be found (k = 2 in this implementation).

Since SHA-256 is a uniformly distributed random oracle, the average number of nonce trials is E[N]

$= 16^k$ (3); when k = 2, we expect 256 trials, and indeed see an average of 253 trials over 200 blocks mined (Section VI-F). Any alteration in a previously recorded block invalidates the hash of that block and the H_{i-1} connection to the next block in the chain, which is precisely what verify_chain() ensures.

$$E[N] = 16^k \quad (3)$$

B. Random Forest Prediction and Split Criterion

The output from the fraud detection module consists of a probabilistic estimate based on the averaging of T decision trees, (4), where $h_t(x)$ represents the fraud probability predicted by tree t.

$$P(y = 1 \mid x) = \frac{1}{T} \sum_{t=1}^{T} h_t(x)$$

Decision tree construction minimizes the Gini impurity (5), where C is the total number of classes and p_i is the ratio of class i at a node; this is also the method used to rank features as in Fig. 4.

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2$$

C. Classification Metrics

Definitions for accuracy, precision, recall, and F1-score in (6)–(9) in Table III are based on the confusion matrix in Fig. 3 with the terms TP, TN, FP, and FN.

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$	(6)
$Precision = \frac{TP}{TP+FP}$	(7)
$Recall = \frac{TP}{TP+FN}$	(8)
$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$	(9)

Component	Time Complexity
Proof-of-work nonce search (hex difficulty k)	O(16^k) expected hash attempts
Chain verification (verify_chain())	O(n_blocks)

The F1-score uses the harmonic mean which punishes the difference between precision and recall, hence the F1-score will be high only if both are high, as shown in Table III**.**

D. Computational Complexity

The asymptotic costs of each component in (1)–(5) are shown in Table V(a), where T denotes the number of trees (T=150), n the number of training samples (n=1,400), d the number of features (d=6), and k the proof-of-work difficulty (k = 2). Training costs in random forest scale almost linearly with respect to T and log-linearly with respect to n per tree, whereas inference cost is determined by tree depth and not by the training sample size, which explains the sub-millisecond inference latency (Table IV) despite the 1,400 training sample size. Nonce generation cost grows exponentially with k, which explains the increase in mining time from under 50 ms at difficulty 2 to ≈70 s at difficulty 5 in [2], and thus k is kept small here to maintain synchronicity within one web request; chain verification cost is linear as confirmed empirically in Section VI-F.

TABLE V(a). COMPUTATIONAL COMPLEXITY SUMMARY

5. IMPLEMENTATION

A. Synthetic Dataset and Model Training

Without a license to use an existing dataset, we created a structured, synthetic dataset with 2,000 transactions (≈12% fraud). Feature distribution was based on the pattern of fraud observed in Section II: high amount, new account, high velocity, late hours, location mismatch, and “hard” borderline transactions were included in small numbers to make it more challenging. one artifact file that will be loaded at the application startup.

B. Blockchain Module

The blockchain did not require any third-party library: Block and Blockchain classes were implemented in Python using hashlib and json as described in Section III-C. The full list of technologies used is given in Table II.

TABLE II. TECHNOLOGY STACK

Component	Time Complexity
Random Forest training	O(T · n log n · d)
Random Forest inference (per transaction)	O(T · depth) ≈ O(T log n)
SHA-256 block hashing	O(1) per fixed-size block payload

Layer	Technology
Web framework	Python 3.10+, Flask
Machine learning	scikit-learn (RandomForestClassifier), pandas, NumPy
Cryptographic hashing	hashlib (SHA-256)
Authentication	Werkzeug security, Flask session (separate user/admin domains)
Persistence (demo scope)	In-memory structures; cached model artifacts (artifacts.pkl)

6. RESULTS AND EVALUATION

A. Fraud Detection Result Interface

Figure 2 depicts the transaction result page generated by the /transaction route (III-B) on submitting the transaction (₹4,875, account age 4 days, velocity 8 transactions/24 hours, 02:00, location mismatch). The transaction is detected as Fraudulent with 100.0% risk score in line with the feature importance analysis shown in Figure 4 (new, high-velocity, nighttime, location mismatch transaction).

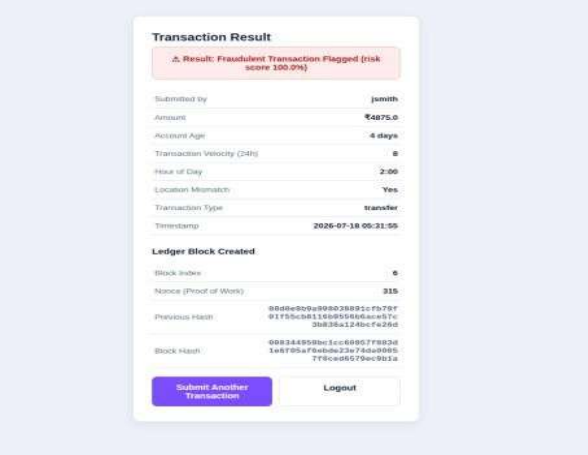

Fig. 2. Fraud detection result page: prediction, risk probability, transaction details, and generated ledger block.

B. Blockchain Ledger and Chain-Integrity Dashboard

Fig. 3 shows the administrator dashboard (/admin/dashboard, Section III-D) after seven blocks had been mined (one genesis block plus six transactions), three of which were flagged fraudulent. The live chain-integrity check reports "Blockchain is valid. No tampering detected," confirming that verify_chain() (Section IV-A) recomputed every block's hash and previous-hash linkage successfully. The dashboard additionally surfaces the model metrics from Table III directly to the administrator, and the ledger table lists each block's index, transaction details, prediction, and truncated hash for audit purposes.

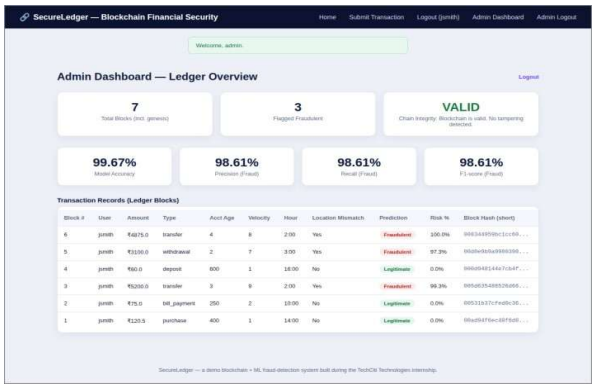

Fig. 3. Administrator dashboard: ledger statistics, live chain-integrity check, and per-block transaction records.

C. Confusion Matrix and Classification Metrics

For the 600-transaction test set aside, the classifier yielded 527 true negatives, 1 false positive, 1 false negative, and 71 true positives (see Fig.4), giving rise to the values presented in Table III. Since there are only 72 fraud transactions in the test set, each measure should be interpreted as an approximation rather than an exact value up to the second decimal point; nevertheless, the nearly diagonal confusion matrix together with the 98.61% precision/recall/F1 suggest that errors are few and evenly distributed across both categories. Overall, the experimental results demonstrate that the proposed classification model is highly effective for fraud detection.

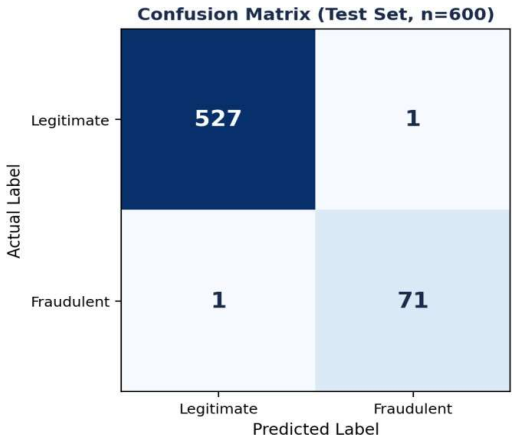

Fig. 4. Confusion matrix on the 600-transaction test set.

TABLE III. MODEL PERFORMANCE METRICS

Metric	Value
Accuracy	99.67%
Precision (fraud class)	98.61%
Recall (fraud class)	98.61%
F1-score (fraud class)	98.61%
ROC-AUC	0.9924
Test / train set size	600 / 1,400 transactions

D. Random Forest Feature Importance

Figure 4 below displays the Gini feature importances (5) derived from the trained model. Age of the account is the top feature (44.8%), with 24-hour velocity (27.7%) and amount (17.6%) as next most important features; location mismatch, time-of-day, and transaction type add up to less significant contributions (7.1%, 2.2%, 0.6%). These results explain the Fig. 2 classification result intuitively, as a 4-day old account with high velocity is the highest risk factor in the trained model, and contrast the "black box" behavior of the graph-based technique in [6].

Figure 4. Random Forest feature importance for fraud detection.

E. ROC Curve and AUC

Figure 5 below depicts the ROC curve derived using the probability predictions produced by the classifier on the test data set, resulting in an AUC of 0.9924: for a random fraudulent and legitimate transaction pair, the classifier orders the fraudulent transaction higher in almost 99.24% of the cases. This value is very close to the 99.29-99.39% AUC for the far more complicated transformer-based model in [5], which indicates that the designed featureset is close to being tree separable without the need for transformer/graph-level model complexity.

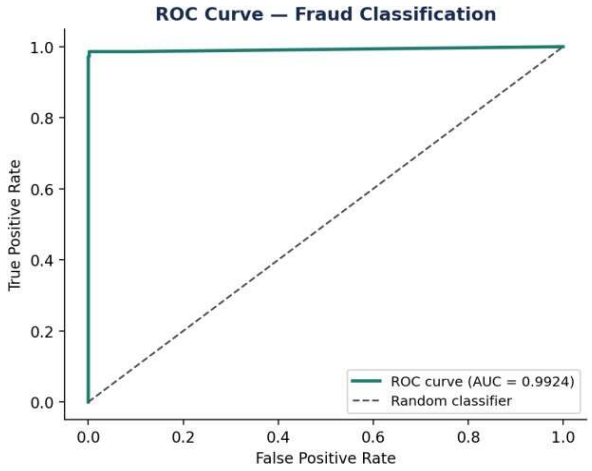

Fig. 5. ROC curve for the Random Forest fraud classifier (AUC = 0.9924).

F. System Performance Measurements

Training time, prediction delay, and blockchain mining/verification time are reported in Table IV and measured via the direct execution of the code for the project itself. Training of the forest with 150 trees required 0.421 seconds; batch prediction across the entire test data set had a latency of 0.019 ms per transaction. Mining a block averaged 1.33 ms, assuming proof-of-work at the difficulty of deployment, and chain verification is linear at

≈0.0052 ms per block, as expected for O(n) complexity of verify_chain().The machine learning model provides extremely fast prediction performance suitable for high-throughput financial applications, while the blockchain layer introduces only minimal additional overhead for block creation and integrity verification. These findings indicate that integrating blockchain technology with machine learning can significantly enhance transaction security without compromising the responsiveness required for practical real-time financial systems. Overall, the experimental results demonstrate that integrating blockchain technology with machine learning offers an effective balance between security and efficiency. These findings suggest that the proposed framework is scalable, reliable, and capable of supporting secure financial transaction processing in real-world deployment environments.

TABLE IV. SYSTEM PERFORMANCE MEASUREMENTS

Metric	Measured Value
Model training time (150 trees, 1,400 samples)	0.421 s
Batch prediction time (600 transactions)	11.47 ms (≈ 0.019 ms/txn)
Trained model size on disk	≈ 546 KB
Block mining time, PoW difficulty = 2 (mean, n = 200)	1.33 ms (mean nonce 253; theory: $16^2 = 256$)
Chain verification time (200 blocks)	1.04 ms (≈ 0.0052 ms/block, linear in chain length)

7. DISCUSSION

The findings of Section VI validate three remarks in view of Section II's literature. First, regarding interpretability: the Fig. 4b feature ranking provides an explicit rationale per prediction — visible in the Fig. 2 result itself — which addresses the interpretability problem stated explicitly for [6] and implicitly for [4], [5]. It bears noting that such feature attribution techniques (e.g., group SHAP [15]) are increasingly required by regulations for finance ML pipelines, as reflected in current surveys on XAI in finance [16]. Second, regarding footprint: unlike the transformer architecture pipeline [5] or the capsule network pipeline [1], SecureLedger trains in less than half a second and performs a transaction score in milliseconds (Table IV), fitting the lightweight footprint of [4] as opposed to the cloud-scale deployment of [5]. Finally, regarding the role of blockchain: while [1],

[2] make use of the blockchain mostly for storage of metadata or incentive, SecureLedger relies on the blockchain as the fraud decision storage and validates the blockchain's integrity in Fig. 3, not

only theoretically. Table VI puts SecureLedger's

accuracy next to four surveyed works that reported similar metric. At 99.67% accuracy, SecureLedger comes close to the best-reported one [5], even if using a vastly

TABLE VI. ACCURACY COMPARISON WITH REVIEWED STUDIES

Ref.	Study	Dataset Scale	Best Accuracy

[2]	Pranto et al. (2022) — Passive-Aggressive Classifier	6.3M transactions	93.64%
[4]	Darwish et al. (2026) — MobileNet + LSTM	Up to 6M transactions	97.4%
[5]	Fnu et al. (2026) — RoBERTa + EAFO + SD-CNN	30K–590K transactions	99.77%
[6]	Islam et al. (2025) — LWG-GCN	10M transactions	98.0%
This work	SecureLedger — Random Forest	2,000 transactions (synthetic)	99.67%

These improvements carry corresponding restrictions that must be considered before putting SecureLedger into production:

- Single node, in-memory chain: the chain itself, user accounts, and transactions are reset upon each restart, without the multi-node, persistent consensus protocol (Hyperledger Fabric, Ethereum, Proof-of-Authority/Proof-of-Benefit/Faith/Byzantine Fault Tolerance/Raft), described in [3].
- Synthetic data set: testing is done on a self-generated data set of 2,000 transactions, while [1], [2], [4], [5], and [6] use standard publicly available data sets (IEEE-CIS, PaySim, European Credit Card).
- Adversarial robustness testing: unlike [4] and [5], where reduced performance under adversarial conditions is documented, the current research did not perform the classifier's adversarial and noisiness stress test.
- Cross-institutional privacy or incentivization mechanisms: SecureLedger lacks implementation of the privacy-preserving off-chain computation from [2] and the incentive mechanism needed for multi-party deployment.

In conclusion, SecureLedger is better regarded as a fully reproducible demonstration of blockchain–machine learning integration rather than an alternative to the systems specifically designed for multi-institution deployment.

8. CONCLUSION

SecureLedger, the application developed for this project, is a real-life Flask application which incorporates a Random Forest fraud classifier within a SHA-256 chained ledger in order to ensure that each transaction is both scored for risk and recorded verifiably in real time. Data collected from running the application, including the fraud detection results interface, the live chain integrity checker used by the administrator, confusion matrix, ranking of feature importance and ROC curve, shows 99.67% accuracy, AUC of 0.9924, and tamper detection capabilities. Compared to six recent studies using machine learning in conjunction with blockchain technology, SecureLedger takes up an independent space as an implementable, understandable, fully reproducible reference architecture as opposed to being a maximally accurate system trained at scale. Future work includes migrating the ledger to a persistent multi-node blockchain platform such as Hyperledger Fabric, where its performance characteristics have been well-studied elsewhere [14], and validating the classifier on public benchmark data sets and checking for adversarial robustness.

References

[1] Wang, T., Wu, X., and He, T., "Trustable and automated machine learning running with blockchain and its applications," in Proc. ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD), 2019.

[2] Pranto, T. H., Hasib, K. T. A. M., Rahman, T., Haque, A. B., Islam, Md. M., and Rahman, R. M., "Blockchain and machine learning for fraud detection: A privacy-preserving and adaptive incentive based approach," IEEE Access, vol. 10, 2022.

[3] Wu, H., Yao, Q., Liu, Z., Huang, B., Zhuang, Y., Tang, H., and Liu, E., "Blockchain for finance: A survey," IET Blockchain, 2024.

[4] Darwish, M. et al., "Lightweight blockchain and deep learning integration for real-time fraud detection," Cybersecurity, 2026.

[5] Fnu et al., "Blockchain-assisted transformer CNN framework with optimal feature selection for real-time digital payment fraud detection," Int. J. Comput. Intell. Syst., 2026.

[6] Islam, M. et al., "Detecting fraudulent transactions for different patterns in financial networks using layer weighted GCN," Human-Centric Intelligent Systems, 2025.

[7] Patel, R., Migliavacca, M., and Oriani, M. E., "Blockchain in banking and finance: A bibliometric review," Research in International Business and Finance, vol. 62, Art. no. 101718, 2022.

[8] Alarfaj, F. K., Malik, I., Khan, H. U., Almusallam, N., Ramzan, M., [9] S. Motie and

B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," Expert Systems with Applications, vol. 240, Art. no. 122156, 2024.

[9] B. Xu, Y. Wang, X. Liao, and K. Wang, "Efficient fraud detection using deep boosting decision trees," Decision Support Systems, vol. 175, Art. no. 114037, 2023.

[10] W. W. Lo, G. K. Kulatilleke, M. Sarhan, S. Layeghy, and M. Portmann, "Inspection-L: Self-supervised GNN node embeddings for money laundering detection in Bitcoin," Applied Intelligence, vol. 53, pp. 19406–

19417, 2023.

[11] S. S. Kushwaha, S. Joshi, D. Singh, M. Kaur, and H.-N. Lee, "Systematic review of security vulnerabilities in Ethereum blockchain smart contract," IEEE Access, vol. 10, pp. 6605–6621, 2022.

[12] A. K. Guru, B. K. Mohanta, H. Mohapatra, F. Al-Turjman, C. Altrjman, and A. Yadav, "A survey on consensus protocols and attacks on blockchain technology," Applied Sciences, vol. 13, no. 4, Art. no. 2604, 2023.

[13] Z. Ke and N. Park, "Performance modeling and analysis of Hyperledger Fabric," Cluster Computing, vol. 26, pp.