*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661, p-ISSN: 2278-8727, Volume 27, Issue 4, Ser. 3 (July. – August. 2025), PP 32-45*
*www.iosrjournals.org*

# Designing A Real-Time, AI-Driven Communication Layer For Fraud Detection In Financial Services

Femi Oke[c], Oyeneye Bolaji, Michael Umakor And Dopamu Oladipupo

***Abstract***
***Introduction***
*Financial fraud in banking and neobanking is a growing threat that demands innovative, real-time defenses. This paper proposes a unified system architecture for fraud detection that combines machine learning-based anomaly detection with a real-time communication (RTC) verification layer and a blockchain-backed audit log. The goal is to detect suspicious transactions using AI, instantly verify potential fraud via video/voice channels, and immutably record all events for transparency. We survey existing work in fraud detection for banking, including the use of Web Real-Time Communication (WebRTC) for remote identity verification, blockchain technology for audit trails, and explainable AI techniques in FinTech. We then present a comprehensive system design, with diagrams illustrating how an AI-driven fraud detection engine can trigger immediate customer verification through WebRTC and log incidents on a blockchain ledger. A methodology is outlined for building or simulating the system using tools like TensorFlow for modeling (e.g., Isolation Forest, LSTM networks), WebRTC frameworks for communication, SHAP for explainability, and Ethereum/Hyperledger for the audit log. We also propose an evaluation plan for benchmarking fraud detection accuracy, system latency, user experience, and audit integrity. The paper concludes with a discussion on the benefits of this multi-layered approach and future work, such as enhancing model explainability and extending the framework to broader financial security use cases.*

---

Date of Submission: 07-08-2025 Date of Acceptance: 17-08-2025

---

## I. Introduction

Financial fraud represents a critical and widespread problem in the banking sector, threatening both institutions and customers. Recent industry surveys underscore the scale of the challenge – for example, a 2022 PwC report found that 56% of companies globally experienced some form of fraud. Fraud not only causes direct financial losses but also erodes customer trust and can destabilize financial systems. Neobanks (digital-only banks) are particularly at risk and under pressure to respond quickly; regulators now expect these institutions to detect and act on suspicious activities instantly to protect consumers and the financial system. Traditional rule-based fraud prevention methods, such as manually defined transaction limits or static heuristics, struggle to keep pace with adaptive fraud tactics and large-scale digital transactions. This gap has driven the financial industry toward advanced **artificial intelligence (AI)** solutions for fraud detection.

AI-driven fraud detection in banking involves applying machine learning (ML) algorithms to analyze vast datasets of transactions and customer behaviors. By analyzing large volumes of data, AI models can learn to distinguish suspicious activities from legitimate transactions, often catching complex fraud patterns that human analysts might miss. Modern banks and fintech companies are integrating such AI systems into their workflows to improve real-time decision-making and fraud prevention. However, purely automated systems can raise new challenges: AI models are not infallible, and they may produce **false positives** (flagging legitimate customer actions as fraud) which negatively impact customer experience. Therefore, a balance must be struck between rapid automated detection and careful validation of alerts to avoid alienating customers with unnecessary account blocks or verification steps.

**Motivation:** This paper is motivated by the need for a *holistic fraud detection architecture* that identifies fraud in real-time and immediately engages the customer (or a fraud analyst) through a secure communication channel for verification, and logs all relevant events in an immutable audit trail. Such a multi-layered approach addresses several pain points: the AI-driven detection provides speed and scalability in spotting anomalies; the real-time communication layer (using technologies like WebRTC) adds a human-in-the-loop verification to reduce false positives and authenticate user identity; and the blockchain-based audit log ensures transparency and trust, which are crucial for compliance and post-incident analysis. By combining these components, the system aims to prevent fraud more effectively while maintaining user trust and meeting stringent regulatory requirements for security, **Know Your Customer (KYC)**, and auditability.

The rest of this paper is organized as follows: Section II reviews related work in fraud detection for banking, the use of WebRTC for identity verification, blockchain for audit trails, and explainable AI in financial services. Section III introduces the proposed system design, including architectural diagrams that illustrate the integration of ML-based fraud detection with a real-time communication verification layer and a blockchain audit log. Section IV outlines the methodology for building or simulating the system, detailing the datasets, tools, and AI models (such as Isolation Forest and LSTM) that could be employed. Section V proposes an evaluation plan for the system's performance. Finally, Section VI concludes the paper and discusses future research directions.

## II. Related Work And Background

### A. Fraud Detection in Banking with AI

Financial institutions have extensively explored machine learning to detect fraud across domains such as credit card transactions, online banking, and insurance. Classical statistical techniques and supervised ML methods (e.g., logistic regression, decision trees, support vector machines) have been applied to identify fraudulent transactions by learning from historical data. In recent years, more sophisticated models, including ensemble methods and deep learning, have shown superior performance in fraud detection tasks. Deep learning (DL) models like convolutional neural networks (CNNs) and recurrent neural networks (RNNs) (particularly **Long Short-Term Memory (LSTM)** networks) can capture complex, nonlinear patterns in large transaction datasets. These models excel at detecting subtle temporal sequences or correlations that rule-based systems might overlook. For instance, Gandhar *et al.* (2024) developed a DL-based approach for transaction anomaly detection that achieved a high F1-score (0.93) on credit card fraud data, outperforming traditional models, and significantly reduced false positives. Such results indicate that AI can not only improve detection accuracy but also minimize "alarm fatigue" by better distinguishing true fraud from normal behavior.

Unsupervised learning and anomaly detection techniques are also vital given the often imbalanced nature of fraud data (where fraudulent instances are rare). **Isolation Forest (IForest)** is one popular anomaly detection algorithm in this domain. IForest operates by randomly partitioning data and isolating outliers with fewer splits than normal instances, effectively identifying anomalies without requiring labeled examples. Prior studies have found that Isolation Forest can achieve high fraud detection accuracy on large datasets while maintaining low false positive rates, making it attractive for real-time fraud screening where labeling every new fraud pattern is impractical. Banks often deploy a combination of *deterministic rules* (for immediate sanity checks) and ML models. A typical workflow might apply instant rule-based checks (e.g., verifying if a transaction exceeds a threshold or violates known fraud patterns) to avoid needless delays, then run the transaction through an ML model for anomaly scoring. If the model flags the transaction as suspicious, further actions are taken before finalizing it.

Despite their power, AI models in finance face scrutiny regarding **transparency** and **fairness**. Black-box models can raise concerns for regulators and banking compliance officers who need to understand why a transaction was flagged or declined. This has spurred interest in **Explainable AI (XAI)** techniques within fraud detection (discussed further in subsection D). In summary, AI has become an indispensable tool in banking fraud detection, enabling real-time analysis of vast transaction streams and adaptive learning of new fraud patterns. The challenge lies in integrating these models into operational processes in a way that balances security with customer convenience and regulatory compliance.

### B. Real-Time Communication for Verification (WebRTC and Video KYC)

Real-time communication technologies have emerged as a crucial component in modern fraud prevention and customer authentication strategies. **Web Real-Time Communication (WebRTC)**, a set of protocols and APIs for peer-to-peer audio and video communication, enables banks to engage with customers instantly within web or mobile applications. One major use case is **Video KYC (Know Your Customer)** and remote identity verification. Rather than relying solely on one-time passwords or out-of-band phone calls, financial institutions are increasingly leveraging video calls to confirm a user's identity in real time. For example, when a high-risk or unusual transaction is initiated, the bank can prompt an immediate video chat with the account holder to verify the transaction's legitimacy. This face-to-face interaction (conducted via WebRTC) allows the bank to visually confirm the person's identity, ask security questions, or observe any suspicious behavior, thereby adding a strong layer of fraud defense. WebRTC-based video calling has the advantage of being embedded directly into the bank's app or website, offering seamless user experience without requiring external software.

Recent industry reports highlight the growing adoption of video calls in banking and fintech. The COVID-19 pandemic accelerated digital onboarding and remote banking services, making video interactions more common and accepted by customers. Regulatory bodies in some jurisdictions (e.g., India's Reserve Bank) have even provided guidelines for video-based KYC, underscoring its importance in secure customer onboarding and authentication. The **fraud prevention** benefits of real-time video verification are significant: it can deter impostors and stop *account takeovers* by requiring a live appearance, and it provides an opportunity to catch

warning signs (such as inconsistent facial features compared to ID documents, or hesitation in answers) that automated systems might miss. Additionally, video sessions can be recorded (with user consent and in compliance with privacy laws) to create an audit trail or for subsequent analysis using AI, for instance to detect behavioral red flags or to train models for fraud pattern recognition in social engineering scams.

Beyond video, the real-time communication layer could also include automated chatbots or voicebots powered by AI to handle initial verification questions. However, video provides a richer context (visual and audible) and is harder for fraudsters to spoof when combined with liveness detection and biometric checks. Some banks have integrated AI services with WebRTC to automate parts of the identity verification; for instance, using OCR (Optical Character Recognition) to read ID documents shown on camera and facial recognition to match the face with the ID photo in real time. This fusion of WebRTC and AI can streamline customer verification while reducing the chances of fraud and human error.

In summary, a real-time communication layer, particularly using WebRTC for video calls, serves as an **AI-driven verification checkpoint**. When the ML-based fraud detection engine flags a transaction, the system can initiate an immediate video call with the user. This not only helps confirm whether the activity is genuine but also serves as a customer engagement tool – honest customers appreciate the proactive security, while fraudsters are often scared off by the requirement to show up on camera. Thus, the RTC layer greatly complements the automated fraud detection by adding human intuition and interaction into the loop at critical moments.

**C. Blockchain for Audit Trails and Financial Integrity**

Blockchain technology, known for underpinning cryptocurrencies, has found compelling applications in enhancing security and transparency in financial services. One such application is the maintenance of *tamper-proof audit trails* for transactions and system actions. A **blockchain** is a distributed ledger where each record (or "block") is cryptographically linked to the previous one, creating an immutable chain of data entries. In the context of fraud detection and financial auditing, using a blockchain-based ledger to log events can ensure that once a transaction or alert is recorded, it cannot be altered or deleted without detection. This immutability is invaluable for forensic analysis and compliance, as it guarantees the integrity of the log data.

Financial institutions are exploring blockchain to bolster **compliance and fraud prevention efforts**. For example, blockchain's audit trail feature provides a transparent record of all transactions and pertinent events, which can be quickly traced to their origin. This capability helps investigators and regulators identify fraudulent activities and understand the sequence of actions that occurred in a fraud incident. In our proposed system, every significant event – such as a model's fraud alert, the initiation of a verification call, the outcome of that call (user confirmed or transaction blocked) – would be written to an audit ledger on a permissioned blockchain network. Because all participants (e.g., the bank's various systems or consortium members, if multiple banks are involved) share the same ledger, it builds trust that the fraud detection process is traceable and **tamper-proof**.

Prior research has highlighted the synergy between AI-based detection and blockchain for trust. Dunsin *et al.* (2021) propose an integrated framework where AI-driven fraud identification is combined with blockchain logging to create a closed-loop system. In their architecture, raw data flows through an AI engine to flag fraud, and each alert triggers a blockchain transaction (potentially via smart contracts) that not only records the alert but can also automate follow-up actions in a **trustless and verifiable manner**. This ensures end-to-end traceability: from data ingestion to fraud decision to response, every step is documented on an immutable ledger. Such an approach can increase accountability of automated decisions and simplify compliance reporting, since regulators could be given access to the blockchain audit trail to independently verify that due process was followed for each incident.

Beyond audit trails, blockchain can offer additional fraud prevention benefits. Its decentralized nature means there is no single point of failure for an attacker to alter records. Smart contracts (self-executing code on the blockchain) can be used to encode business rules or alerts – for instance, automatically flagging and halting a transaction that appears in multiple banks' fraud ledgers or that violates certain predefined criteria. In an insurance fraud example, a blockchain can prevent double-claiming by ensuring that once a claim is recorded by one company it's visible to others. Similarly, in banking, a consortium of banks could share a blockchain for recording confirmed fraud cases, making it harder for serial fraudsters to jump from one institution to another (though privacy and data protection issues would need to be managed).

For the scope of this paper, the focus is on using blockchain internally (or within a consortium) as a robust audit log for the fraud detection and response process. The **immutable ledger with time-stamped entries** guarantees data integrity and can be designed to include cryptographic hashes of transaction details or model outputs for later verification. As noted in a FinTech study, blockchain audit trails greatly enhance transparency and reduce fraud by providing a single source of truth for transactions and checks. This aligns well with financial regulatory requirements for maintaining detailed records of suspicious activities (e.g., under AML regulations). By leveraging blockchain in our system, we aim to instill confidence that no fraudulent alert or response can be

retrospectively manipulated, and that a permanent record exists to learn from incidents and improve the models and rules over time.

**D. Explainable AI in FinTech**

As financial institutions deploy AI models for high-stakes decisions, such as fraud detection, the need for explainability and transparency in these models has become paramount. **Explainable AI (XAI)** refers to techniques and tools that make the decision-making of AI systems understandable to humans. In the FinTech context, XAI helps bridge the gap between complex model predictions and the requirements of regulators, auditors, and customers to know *why* a decision (such as blocking a transaction) was made.

Regulators worldwide have introduced guidelines (and even regulations, such as the EU's GDPR and proposed AI Act) insisting on a certain level of explainability for automated decisions in finance, especially those related to credit, fraud, or compliance. This is driven by concerns over bias, fairness, and the potential for AI errors. Consequently, banks are incorporating XAI methods into their fraud detection systems to ensure accountability. For instance, *model-agnostic* explanation techniques like LIME (Local Interpretable Model-agnostic Explanations) and **SHAP** (SHapley Additive exPlanations) are frequently used to interpret fraud models. SHAP, based on cooperative game theory, attributes an importance value to each feature for a given prediction, effectively telling which factors pushed the model towards classifying a transaction as fraudulent. According to recent research, integrating SHAP with fraud detection models can improve regulatory compliance and financial transparency. Thanathamathee *et al.* (2024) demonstrate that a SHAP-enhanced ensemble model not only detects credit card fraud effectively but also produces explanations that satisfy auditors by highlighting the key risk factors in each flagged case.

Explainability is not just for regulators; it also aids internal analysts and can even be used to communicate with customers. In a system like ours, when an alert is raised and a WebRTC verification call is triggered, an explanation of the alert could be provided to the bank's fraud officer or even to the customer (in simplified form) to build trust. For example, the system might show that *"this transaction was flagged because it's far outside your usual location and spending amount"*, based on the model's top features. This aligns with the idea that integrating XAI into fraud detection is crucial for ensuring accountability and compliance. XAI techniques can identify which transaction attributes (e.g., IP address, purchase category, time of transaction) most influenced the model's suspicion score. Having this insight allows human reviewers to verify the model's reasoning and catch any potential model mistakes or biases. It also helps in refining the model by revealing which features are highly influential; if those features correspond to policies or rules that can be adjusted, the bank may tweak thresholds to reduce false positives.

A notable challenge is achieving **real-time explainability**. Traditional XAI methods can be computationally intensive (for instance, SHAP can be slow on large models), but recent advancements and approximations are making it feasible to generate explanations quickly enough to be useful in a live fraud prevention system. Some banks use precomputed SHAP values for common patterns, or simpler surrogate models to approximate explanations on the fly. Moreover, having an explanation for each alert in the blockchain audit log could prove valuable; it means that for every fraudulent event or false alarm, there is a recorded rationale, which is ideal for post-mortem analyses and continuous improvement of the AI models.

In summary, XAI in FinTech (and specifically in fraud detection) enhances trust in AI systems by making their decisions more transparent. Techniques like SHAP and LIME help highlight why a transaction was flagged, which is beneficial for compliance and for fine-tuning models to be both effective and fair. Our proposed system incorporates explainability by design: the ML fraud detection engine would generate explanation metadata (e.g., top contributing features to the risk score), which can be reviewed by analysts and stored in the audit log. This ensures that the **AI-driven communication layer** not only acts quickly and intelligently but also does so in a manner that stakeholders can understand and evaluate.

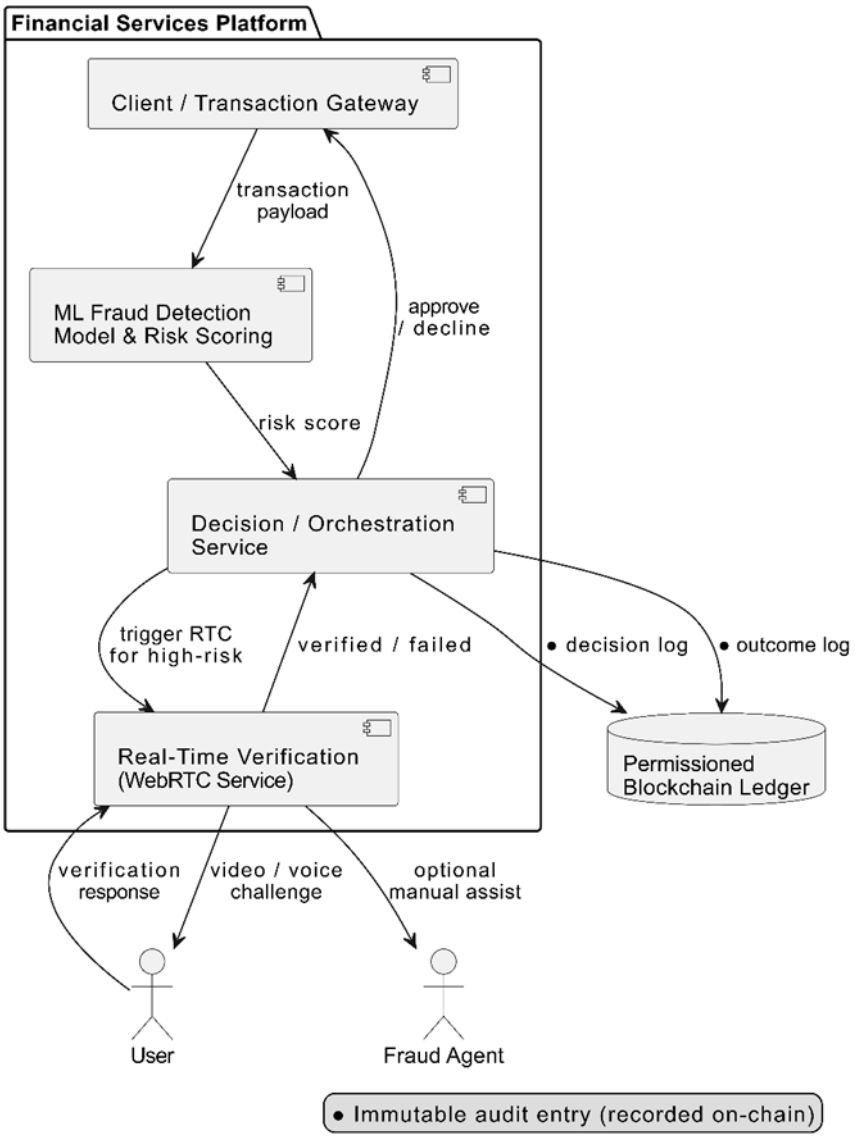
**System Design**

**Overview:** The proposed system is a multi-layered fraud detection and response architecture tailored for real-time financial transactions (typical of banking and neobanking platforms). It consists of three primary integrated components: (1) a Machine Learning-based Fraud Detection Engine, (2) a Real-Time Communication Verification Layer, and (3) a Blockchain-based Audit Log. Figure 1 illustrates the high-level architecture and data flow among these components. The design follows an event-driven paradigm: each transaction triggers a cascade of processing steps – data analysis by the ML engine, a decision on whether human verification is needed, initiation of a communication session if required, and logging of outcomes to the ledger. This design ensures **closed-loop feedback**, meaning that every decision and action is captured for future analysis, and the system can learn from past incidents.

*Figure 1: High-level system architecture integrating AI-driven fraud detection with a real-time communication (RTC) verification layer and a blockchain audit log. The ML Fraud Detection Model processes*

*incoming transactions and assigns a risk score. If a transaction is deemed suspicious, the system triggers an RTC verification (e.g., a WebRTC video call) with the user or a fraud agent. All decisions, alerts, and outcomes are recorded on an immutable blockchain ledger for audit and compliance purposes.*

**Figure 1 - AI-Driven Fraud Detection with RTC Verification & Blockchain Audit**




**Fraud Detection Engine:** At the core of the system is the fraud detection engine, powered by one or multiple ML models. This engine ingests streaming transaction data and contextual information (such as user profile, device data, historical transaction patterns) in real time. It outputs a **risk score** or classification for each transaction. A low score indicates the transaction is likely legitimate, whereas a high score indicates suspicion of fraud. The engine can be composed of hybrid models to cover various fraud scenarios:

- An **Isolation Forest model** may continuously monitor for anomalous patterns that deviate from a customer's normal behavior, without relying on labeled fraud examples.
- A supervised **classification model** (e.g., gradient boosted trees or neural network) might be trained on historical fraud cases to predict the probability that a transaction is fraudulent.
- A sequential model like an **LSTM network** could analyze sequences of actions (e.g., multiple transfers in a short time, or login location changes) to detect complex fraud MOs (modus operandi) that unfold over time.

The engine could also utilize ensemble methods or stacking (combining multiple model outputs) to improve robustness.

To meet real-time requirements, this detection engine is deployed in a streaming data processing environment. Transaction events from the bank's systems feed into the model inference service with minimal latency. If the **risk score** exceeds a predefined threshold, the engine flags the transaction as *suspicious*. In many banks today, such flags would either automatically decline the transaction or queue it for manual review. In our design, a suspicious flag triggers the next layer – the communication verification – instead of outright denying the transaction. This balances security with user convenience, only escalating to a human step when truly necessary. Notably, before invoking the user verification, the system can perform a final check using simple business rules (for example, blocking if the amount is extremely high regardless of user confirmation, or if multiple rapid attempts are detected, etc.) – these rules can be implemented as smart contract conditions in the blockchain as well.

**Real-Time Communication Verification Layer:** This layer is activated for transactions flagged by the ML engine. The system establishes a secure **WebRTC** connection to either the customer's device or a fraud investigation agent (or both in a conference) for live verification. For the customer, the bank's mobile app or web portal would instantly prompt a video call, possibly with a message like "Unusual activity detected – please verify this transaction." Through the video call, the customer can confirm if they initiated the transaction. They may be asked to show ID or simply respond to security questions. The bank representative or AI-driven assistant on the other end can observe and validate the user's identity and intent. This step serves as a real-time challenge-response mechanism. If the customer confirms the transaction and passes the identity check, the transaction can be completed. If the customer denies making the transaction or fails the verification (e.g., cannot adequately prove identity), the transaction is halted and flagged as fraud.

In addition to video, the communication layer could incorporate one-time passcodes or push notifications for lower-risk cases. However, video/voice provides higher assurance. The layer should be designed to minimize user friction – for example, it should automatically route the call to an available fraud desk agent if a human is needed, or utilize an AI voicebot to ask initial questions if human staff are busy. Thanks to WebRTC's low-latency peer-to-peer nature, this verification can occur within seconds of the transaction attempt, thereby truly operating in real time. One of the benefits of this approach is that it also provides a **personal touch** to fraud prevention. Rather than an unexplained decline, the user is engaged and aware of the security process, which can increase trust in the institution's vigilance.

From a system perspective, the decision outcomes from this layer are fed back into the pipeline:

- If the transaction is verified as legitimate (false alarm by the model), the system marks it as safe and allows it to proceed. This outcome can be used as a feedback signal to the ML model (for example, to retrain or adjust thresholds to reduce future false positives).
- If the transaction is confirmed fraudulent or unverified, the system blocks it and may take further actions (freezing the account, alerting the fraud department, etc.). This too is valuable feedback for model improvement and triggers downstream processes (like filing a Suspicious Activity Report, SAR, if required by law).

Importantly, the communication session itself can be recorded and analyzed. Advanced fraud prevention setups might run real-time face recognition or behavior analysis during the call – for instance, verifying that the face matches the customer's ID on file, or analyzing voice stress levels to detect potential coercion. These AI enhancements align with trends of combining AI and WebRTC for security. In any case, the RTC layer acts as a critical gate that significantly raises the bar for fraudsters. It's relatively easy for a fraudster to script or automate fraudulent transactions, but much harder to fool a live video verification that involves biometric identity confirmation and unpredictable human interaction.

**Blockchain Audit Log:** Every action and decision in the above process is recorded on a permissioned blockchain ledger to ensure an immutable audit trail. The blockchain component could be implemented using a technology like **Hyperledger Fabric** (which is suited for private consortia) or even **Ethereum** (either the public network or a private instance) with smart contracts. Each relevant event generates a transaction on the blockchain:

- When the ML engine flags a transaction, a log entry is created (containing a hash or ID of the transaction, the risk score, timestamp, and model identifier).
- When a verification call is initiated, another entry logs this event (including time, the parties involved – perhaps referenced by an anonymized ID – and the transaction ID it relates to).
- After verification, the outcome (verified genuine, or confirmed fraud) is logged, along with any relevant metadata (e.g., an analyst's ID if a human made the final call, or a short code for why it was deemed fraud).

By chaining these events on blockchain, we create a chronological, tamper-proof record of the fraud detection and response lifecycle for each case. This has multiple benefits. First, it aids in **compliance and reporting**: auditors can review the blockchain log to ensure the bank's fraud response followed approved procedures (for instance, that a transaction above a certain amount always triggered the required verification

steps). Second, it provides forensic evidence in the event of disputes – if a customer claims a transaction was wrongly approved or declined, the bank has an immutable record of the checks performed and the customer's response. Third, it enhances inter-bank collaboration: if this system were extended to a network of banks, they could share a blockchain ledger (or interoperable ledgers) of confirmed fraud incidents. This would help in catching cross-institution fraud (e.g., fraudster moves from Bank A to Bank B) as mentioned in some studies. In our primary use case (within one banking organization), the blockchain ensures trust internally between components and over time – even administrators cannot secretly alter the logs to cover up a failure in the process, due to the blockchain's **immutability and consensus** mechanisms.

The blockchain's smart contract logic can also automate parts of the workflow. For example, a smart contract could define that "if a suspicious transaction is logged and no verification outcome is logged within 5 minutes, auto-freeze the account." This creates a fail-safe in case the communication layer fails or is circumvented. Another smart contract could automate alert notifications to regulators if certain thresholds are hit (e.g., a very large fraud attempt). These contracts run on the blockchain network, ensuring their execution is transparent and cannot be skipped. Essentially, the blockchain acts as the **nervous system** connecting the AI brain (fraud model) and the communication muscle (RTC layer), with built-in reflexes (contracts) and memory (ledger).

**Integration of Components:** The three components are designed to work in concert. The ML engine is continuously monitoring and feeding results to both the user-facing systems and the blockchain. The blockchain logs can also feed back to model training – for instance, confirmed fraud cases recorded on the ledger become part of the training dataset for improving the model. The communication layer relies on triggers from the ML engine and in turn provides feedback to both the ML engine and the blockchain. Security and privacy are paramount: all sensitive data written to the blockchain can be hashed or encrypted (with keys held by the bank and authorized regulators) to protect customer information, since blockchains are append-only. Moreover, the real-time nature of the system requires careful optimization: the latency of writing to the blockchain and setting up calls should not be so high that it disrupts the customer experience. Permissioned blockchains like Hyperledger can commit transactions in seconds or less, which is sufficient for our purposes (especially since the user verification itself might take tens of seconds, which dominates the timeline).

In summary, the proposed design creates a layered defense: AI for **detection**, RTC for **verification**, and blockchain for **integrity**. It aims to catch fraudulent transactions that slip past simpler controls, involve the user immediately to confirm or reject flagged transactions (thereby preventing fraud in progress), and maintain an incontrovertible record of all such events for oversight. By focusing on banking and neobanking use cases, the design emphasizes real-time operation, scalability (handling potentially thousands of transactions per second in a big bank scenario), and compliance with financial regulations.

## III. Methodology

In this section, we outline how the proposed system can be built or simulated, detailing the datasets, tools, and methods for each component. The methodology covers development steps for the machine learning model, the integration of the communication layer, the setup of the blockchain ledger, and the implementation of explainability features.

**Data Collection and Datasets:** To develop and test the fraud detection engine, historical transaction data is required. For banks, this typically includes records of legitimate transactions and confirmed fraudulent transactions (labels). In a simulation or academic setting, one could use publicly available datasets such as the **Kaggle Credit Card Fraud Detection Dataset**, which contains real credit card transactions labeled as fraudulent or genuine. This dataset is widely used in fraud detection research and would allow training of models and evaluation of detection performance. Another source could be synthetic banking transaction data generated to mimic a neobank's profile, possibly including user behavior patterns over time for testing sequence models. If focusing on *neobanking*, data might also include various digital channels (mobile payments, peer-to-peer transfers, etc.). Additionally, collecting example data for the communication layer is useful – e.g., logs of previous verification calls (if available) or at least scenarios that describe how users respond. For the blockchain log, there is no training data per se, but we define the schema of data to record (transaction IDs, timestamps, flags, etc.).

**Machine Learning Model Development:** We consider several model types:

- *Isolation Forest:* We would configure an Isolation Forest to train on a sample of normal transactions from the dataset (unsupervised training). Tuning involves setting the number of trees and subsampling size; this model will then produce an anomaly score for each new transaction. Transactions with scores above a threshold (determined by desired false positive rate) are deemed suspicious.
- *LSTM-based Neural Network:* For capturing temporal patterns, an LSTM model can be trained on sequences of transactions. For example, one could transform user transaction history into sequences (sorted by time) and

label sequences that contain a fraud occurrence. The LSTM could be trained to output a sequence-level fraud probability or to flag anomalous sequence elements. Tools like **TensorFlow** or **PyTorch** would be used to design and train the neural network. Hyperparameters like number of LSTM units, sequence length, and training epochs would be selected through experimentation on validation data.

- *Supervised Classifier (e.g., XGBoost):* Using labeled data (fraud/not fraud), we can train a gradient boosted tree model (such as **XGBoost** or LightGBM) using features like transaction amount, time of day, merchant category, user's typical spending profile, device location, etc. These models are often very effective for tabular financial data and provide feature importance that can be later utilized for explanations. During training, techniques to handle class imbalance (since fraud cases are rare) are needed – e.g., oversampling fraud cases, or using appropriate evaluation metrics (ROC AUC, precision-recall) rather than raw accuracy.
- *Ensemble Integration:* The final risk score might be a combination of multiple models. For instance, the system could flag a transaction if **any** model is highly confident it's fraud, or use a weighted average of scores. A simple but effective approach is to calibrate each model's output to a probability and then use a logical OR for triggers (to maximize catching frauds) while recording which model(s) raised the alert for explanation purposes.

All model training would be done on historical data, and then the models would be deployed for live inference. We must ensure the inference pipeline is optimized for speed – for example, pre-loading models in memory, possibly using batch processing or streaming frameworks like Apache Flink or Kafka Streams if dealing with extremely high volume. The **verification threshold** for the risk score is a critical parameter; it could be set by analyzing the validation set to find a threshold that yields an acceptable false positive rate (e.g., we may aim to only bother users with calls at a rate of say 1 per 1000 genuine transactions, depending on business tolerance, while still catching most fraud). This threshold might be dynamic, adjusting based on system load or user segment (VIP customers might have a different threshold than new customers, etc., implementing a risk-based approach).

**Real-Time Communication Implementation:** The communication layer can be implemented using WebRTC APIs and a signaling server infrastructure. For simulation, we can assume the existence of a **WebRTC service** (like an open-source SFU – Selective Forwarding Unit, or cloud-based API such as Twilio, Vonage, etc.) to handle the video call setup and streaming. In practice, the bank's mobile app or web app would include WebRTC client code to connect to the bank's server when a verification is needed. For prototyping, one could use an open-source project such as **OpenTok** or **Jitsi Meet** to create a quick video calling solution restricted to the context of verification calls.

We will integrate this by having the fraud detection engine emit an event (perhaps on a message bus) when a transaction is suspicious. A *Verification Service* listens for these events and orchestrates the WebRTC session: it sends a push notification to the user's device to initiate a call, and alerts a bank fraud officer dashboard. If automating, we might create a bot that joins the call to prompt the user until a human can join. For simulation or testing purposes, this could be simplified to logging that a call would have been made and maybe simulating a user response (e.g., assume X% of suspicious transactions are confirmed by user as genuine, and (100–X)% are not).

Key tools and considerations:

- Use of secure communication (WebRTC supports DTLS/SRTP encryption natively for streams, which is essential for privacy).
- Logging the content of the verification (not video itself but the result) to the blockchain as described.
- Timeout handling: if user does not respond within a certain time, the methodology should define what happens (likely fail-safe: block the transaction).
- UI/UX design for the user prompt is also part of implementation (outside the scope of this paper, but crucial in real deployment).

**Blockchain Setup:** We choose a blockchain platform suitable for our needs. **Hyperledger Fabric** is a good choice for permissioned networks and can be run within a bank's IT environment or across a consortium of banks. Alternatively, a private Ethereum network (or Quorum, an enterprise fork of Ethereum) could be used if smart contract expressiveness is needed. For methodology, let's assume Hyperledger Fabric for simplicity, as it provides fine-grained access control and efficient consensus (through RAFT or Kafka ordering service) for a limited set of nodes (e.g., nodes run by the bank's departments or partner regulatory node).

Steps to implement the blockchain audit log:

- Define the data model for the ledger entries. This could be a simple key-value store where the key is a unique event ID (or composite of transaction ID and timestamp) and the value is a JSON object containing: transaction_id, event_type (e.g., "ML_ALERT", "CALL_INITIATED", "CALL_RESULT"), timestamp,

details. The details field can hold additional info like risk score or explanation summary, user ID (hashed), verification outcome, etc.

- Develop chaincode (smart contract in Fabric) that defines transactions for writing these events. For example, LogAlert(tx_id, score, features[]) and LogOutcome(tx_id, outcome) as two functions. The contract can enforce rules, like an outcome can only be logged if a prior alert exists, etc., to maintain consistency.
- Spin up the blockchain network (in simulation, this can be done with a few Docker containers running Fabric peer nodes and an ordering service). Initialize a channel and deploy the chaincode.
- Integrate the application with the blockchain: the fraud detection engine and verification service will act as clients that invoke the smart contract. Using Fabric's SDK (available for languages like Go, Node.js, Python), the system will send transactions to the blockchain for each event. We would test this by generating some events from the fraud engine (either from actual model triggers during a test run or by simulating triggers).
- Verify that the events are being recorded immutably: try querying the ledger for a particular transaction's trail and ensure it matches expected outcomes. Also test unauthorized access – ensure that only permitted roles can write or read the log (Fabric allows setting membership service such that, for example, only the bank and regulator nodes have the keys to read all data).

For simulation, one could simplify by using a local Ethereum node (like Ganache) and writing a solidity contract to log events; however, ensuring immutability and proper access in a realistic way is better with Fabric in the banking scenario.

**Explainability Integration:** During model development, we select an XAI method to accompany the predictions. We will use **SHAP** as a primary method, given its strong theoretical foundation for feature attribution. After the ML model is trained, we compute SHAP values for a representative set of fraud and non-fraud examples to understand the model's global behavior. More importantly, at runtime, whenever the model flags a transaction, we can compute the SHAP values for that specific instance (many ML libraries allow fast SHAP computation if using tree models or by approximating for neural nets). The top features contributing to a "fraud" prediction are then captured. For instance, the model might flag a transaction because "Device location = overseas (contributed +0.4 to risk), Unseen merchant (contributed +0.2), High amount (contributed +0.15)" etc. We package these into an explanation string or object.

The methodology for explainability:

- Use the **SHAP Python library** for tree models or DeepSHAP for neural networks. Validate that the explanations align with domain intuition (e.g., known risk factors are indeed given weight).
- Include the explanation generation in the inference pipeline. This can be optimized by only computing it when needed (i.e., only for transactions above the suspicion threshold, to save computation).
- Log the key explanation results in the blockchain (e.g., include top 3 features and their sign in the details of the ML_ALERT event).
- Optionally, use LIME as a cross-check for a few cases to ensure consistency of explanations.

**Prototype and Simulation:** Bringing it all together, a prototype of the system could be run on a smaller scale:

1. Simulate a stream of transactions (either replaying from a dataset or generating synthetic events). This could be done with a script producing events with random normal transactions and some injected fraud attempts.
2. The ML engine (running in a loop or streaming job) processes each transaction, uses the trained model to score it, and decides to pass or trigger verification.
3. For each trigger, log the alert to blockchain and either simulate a user verification or actually perform one if a user interface is connected. A simple simulation for verification could be: assume, say, 50% of flagged events are true fraud (user denies) and 50% false alarm (user confirms). This can be randomized or predetermined for testing.
4. Log the verification outcome to blockchain.
5. Monitor the output: ensure the number of calls triggered is as expected, and the blockchain ledger shows the sequence of events correctly.

We would measure how quickly the system reacts (time from transaction to user call) to ensure it meets real-time criteria. For instance, if end-to-end from model flag to call ringing is under, say, 5 seconds, that is likely acceptable in practice.

**Tools and Technologies Summary:**

- *ML/DL Frameworks:* **TensorFlow** or **PyTorch** for model training; **Scikit-learn** or **XGBoost** for traditional models. These provide the algorithms (Isolation Forest, neural nets, etc.) and also could be used with SHAP (SHAP has wrappers for XGBoost and deep models).

- *Data Processing:* **Pandas** for offline data prep; **Apache Kafka** or **Apache Flink** for streaming (if demonstrating real-time scoring on streaming input).
- *WebRTC Communication:* Use a library or service that supports WebRTC, e.g., **Node.js** with a signaling server like Socket.io for coordinating calls, and a front-end using the browser's WebRTC API. Optionally, libraries like **PeerJS** could simplify peer connection management. For mobile, use native WebRTC libraries.
- *Blockchain:* **Hyperledger Fabric** (with chaincode in Go/JavaScript), or **Ethereum** (smart contracts in Solidity). For quick iteration, one could even use an in-memory blockchain or a local Ethereum testnet to simulate writes.
- *Explainability:* **SHAP** library, and possibly **LIME** for comparisons. Tools like **IBM's AI Explainability 360** could also be explored if more advanced methods are needed (though SHAP suffices here).

By following this methodology, one can build a functional prototype of the system. The key is ensuring each component works individually (model detects anomalies well, WebRTC can connect and get a response, blockchain logs transactions correctly) and then integrating them with proper messaging between components. Testing would involve various scenarios: genuine user making a normal transaction (should pass through without video call), genuine user making an unusual but legitimate transaction (should trigger call and then be approved), a fraudster attempt (should trigger call, not be verified, thus blocked), and system edge cases (user doesn't answer call, or model false alarm). The simulation results would inform any necessary adjustments to thresholds, model parameters, or timeouts in the communication layer.

**Evaluation Plan**

Evaluating the effectiveness of the proposed fraud detection and communication system is multifaceted. We need to assess not only the **accuracy of fraud detection** but also the system's real-time performance, its impact on user experience, and the reliability of the audit logging. Below, we outline an evaluation plan with key metrics and methods for each dimension:

**1. Fraud Detection Performance:** Using standard metrics from machine learning, we will evaluate how well the AI model (or ensemble of models) identifies fraudulent transactions.

- *Detection Rate (Recall)*: The proportion of actual fraudulent transactions that the system correctly flags. A high recall means most frauds are caught, which is critical for security.
- *False Positive Rate*: The proportion of legitimate transactions that are incorrectly flagged as suspicious. Since each flagged case triggers a user verification, this rate should be kept low to avoid burdening customers and operations.
- *Precision*: Among the transactions flagged, the percentage that were truly fraudulent. High precision indicates that when the system raises an alert, it is usually correct (which is good for efficiency of the verification layer).
- *F1-Score*: The harmonic mean of precision and recall, giving a single measure of accuracy that balances fraud catch and false alarms.
- *ROC/AUC*: The Area Under the Receiver Operating Characteristic curve can be measured to understand the trade-off between true positive rate and false positive rate at various threshold settings. This is useful to pick the operating threshold for the model.

We would compute these metrics on a labeled test dataset (not seen by the model during training). For example, using the Kaggle dataset or a held-out portion of the bank's data. If the dataset is heavily imbalanced, precision-recall curves and the precision at a certain recall (or vice versa) are particularly insightful.

**2. Real-Time System Performance:** Since a goal is to have a *real-time* solution, we must measure the timing and throughput aspects:

- *Latency:* The end-to-end latency from transaction initiation to fraud decision, and if flagged, to verification completion. We can break this down:
  - Model scoring latency (typically in milliseconds, but must be measured under load conditions to ensure the model can keep up with transaction volume).
  - Call setup latency: how long after a flag until the user's device rings or receives the verification prompt.
  - Verification interaction time: time for the user to respond and complete the verification (this can vary widely; for evaluation, we might measure average call duration or time to resolve).
  - Ideally, for a seamless experience, if a transaction is flagged, the user should be notified within a few seconds. We will test the system with various network conditions to ensure the WebRTC call quality and connection time are within acceptable limits. Tools like WebRTC internals or custom logging can capture call setup times and any connection issues.
- *Throughput and Scalability:* Can the system handle the volume of transactions typical for the target environment? For a neobank, this might be thousands of transactions per second at peak. We can stress-test the ML engine with high transaction rates (using simulated input) and observe CPU/memory usage and message

queue backlogs. The blockchain's throughput is also a factor – permissioned blockchains can usually handle a few hundred transactions per second; we must ensure logging each event doesn't become a bottleneck. If needed, some events could be batched or logged asynchronously to the blockchain. The evaluation would record how many transactions per second the system can process before delays increase beyond a threshold.

**3. User Experience and Impact:** Even if a system is accurate, if it frustrates customers, it will not be viable. We evaluate:

- *False Positive Customer Impact:* Using the false positive rate from above, estimate how often an average user might get a verification call unnecessarily. For instance, "on average, one legitimate transaction in 500 triggers a verification call." We can compare this to industry benchmarks or acceptable thresholds determined by the bank's policy. The aim is to minimize these unnecessary challenges.
- *Verification Success Rate:* Among the verification calls made, in how many cases is the transaction ultimately verified as genuine versus confirmed fraud? A high genuine verification rate might indicate the threshold could be raised to be less sensitive; a low rate (most calls uncover fraud) indicates the system is efficiently targeting likely fraud.
- *Drop-off rate:* If possible, measure if any users abandon transactions because of the added step. For example, if a user gets a call and fails to complete it (maybe they hang up or ignore it), what happens? This is more of a qualitative measure but can be gleaned from logs. We want to ensure legitimate users aren't deterred by the security procedure. In a pilot test, one could survey users who experienced a verification call for feedback on whether it felt like a positive security measure or an annoyance.
- *Fraud Loss Reduction:* If we have historical data on fraud losses or fraud throughputs, compare the scenario of using this system vs. a baseline (e.g., previous rule-based system). This might require simulation: take a test set of past transactions where some frauds were missed by older systems and see if our AI+RTC system would have caught them (thus potentially preventing loss). Also consider how quickly fraud cases are resolved now – if a fraud attempt is caught in real time before money leaves the institution, losses can be prevented entirely, as opposed to post-factum investigations.

**4. Security and Robustness:** Evaluate whether the introduction of the communication layer and blockchain has any security loopholes or performance issues:

- *Fraudster Response:* It's hard to quantify, but we can analyze how a fraudster might try to circumvent the system. For example, could they fake or hijack the WebRTC call? We ensure the calls are authenticated (the app should verify it's talking to the real user's device). For evaluation, one might simulate an adversary (with permission) trying to break into a session or observing that without the correct cryptographic keys it's infeasible.
- *Blockchain Integrity:* Validate that the audit trail is indeed immutable. In testing, attempt to alter a record (which should be rejected by consensus). If using Fabric, ensure endorsement policies are set such that no single node can alter records arbitrarily. Also measure the overhead: logging to blockchain vs. a traditional database – e.g., latency added per transaction log. If the overhead is minimal (which we expect if tuned properly), then it's justified for the integrity gain. We might also test resilience: if a node goes down, does the logging continue (blockchain networks typically have fault tolerance with multiple nodes).
- *Explainability Effectiveness:* This is a qualitative but important part of evaluation. We can present a few example fraud alerts with their SHAP explanations to domain experts (fraud analysts) and gauge if the explanations are understandable and helpful. Also test the generation time of explanations – if computing a SHAP explanation for a flagged transaction takes too long (say several seconds), that could slow down the process; ideally, it should be under a second. If too slow, consider simpler explanation techniques (which we could evaluate for fidelity). The output of this part would be a confirmation that the XAI component provides value (e.g., an analyst was able to use it to quickly approve a false positive because they saw it was flagged due to an unusual but explainable feature).

**5. End-to-End Simulation Results:** Combine the above in an end-to-end test scenario. For example, simulate 1000 transactions with a mix of 20 fraud attempts. Measure:

- How many of the fraud attempts were stopped (and how quickly).
- How many verification calls were made (say 30 calls, of which 10 were actual fraud, 20 were false alarms).
- The average handling time per case.
- Any backlog or delay introduced (did some transactions pile up waiting for verification if calls took a long time? This could indicate need for parallel call handling via more agents).
- Resource usage (CPU for ML, network usage for video calls, etc.) to ensure the system is practically deployable on the bank's infrastructure.

By analyzing these results, we can identify bottlenecks or areas for improvement. For instance, if the false positive rate is slightly high, one might adjust the model or threshold and repeat tests. If the blockchain logging is slow, one could adjust block size or frequency of committing in Fabric.

**Benchmarking:** It's useful to compare our integrated approach to a couple of benchmarks:

- A baseline rule-based system (if data available, how many frauds that would catch vs ours).
- An AI-only system without the communication layer (this would have to automatically decline suspicious transactions). We can compare how many genuine transactions that approach would wrongfully decline vs. our approach which would have saved them via verification. This illustrates the value-add of the RTC layer.
- Perhaps compare to a human-only review scenario (not feasible in real-time for all transactions, but maybe historically the bank had a manual review team for large transactions). Metrics like average fraud resolution time can be compared (ours would be minutes or seconds, manual might be hours).

The evaluation phase would conclude with the findings that demonstrate the system's efficacy. We expect to show that the system can significantly improve fraud detection rates while keeping false positives manageable via the verification step, thus reducing fraud losses without undue customer friction. Additionally, we anticipate that the blockchain audit trail adds only minor overhead but greatly enhances the transparency and trustworthiness of the process, which is a favorable trade-off. These results would justify the practicality of deploying such a system in a banking or neobanking environment.

## IV. Conclusion And Future Work

In this paper, we presented a novel design for a real-time, AI-driven fraud detection system augmented with an interactive communication layer and a blockchain audit trail. Focusing on the banking and neobanking use case, we addressed the need for immediate fraud interception, customer verification, and immutable logging to meet both security and compliance demands. The proposed architecture leverages the strengths of machine learning for detecting suspicious transactions, WebRTC-based video calls for on-the-spot identity verification, and blockchain technology for ensuring that every alert and response is transparently recorded in a tamper-proof manner. By integrating explainable AI techniques (such as SHAP) into the pipeline, the system also strives to maintain clarity and trust in decisions, an essential factor in the finance domain.

The design achieves a balance between automation and human oversight: AI catches patterns at scale and speed that humans cannot, while the real-time communication with users adds a human judgment element to confirm or refute the AI's findings, thereby significantly reducing false positives. The blockchain audit layer provides a single source of truth for all fraud-related events, enhancing collaboration among stakeholders (fraud analysts, auditors, regulators) and enabling easier compliance reporting. We illustrated how such a system can be built using current technologies and discussed an evaluation plan indicating its potential to improve fraud detection rates and response times without sacrificing customer experience. In effect, this multi-layer approach acts as a *fraud defense stack* – each layer reinforcing the others to create a more resilient financial platform.

**Future Work:** There are several avenues to extend and refine this work. One direction is to incorporate **advanced AI models** such as graph neural networks (GNNs) for fraud detection. Financial fraud often involves networks of entities (accounts, merchants, devices); GNNs could be used to detect ring activities or money laundering patterns by analyzing transaction graphs. Integrating a GNN module could further improve detection of complex collusive fraud that individual transaction analysis might miss. Another extension is to explore **federated learning** or consortium learning approaches, where multiple banks collaboratively train fraud detection models on shared patterns without directly exchanging customer data – the blockchain ledger of fraud incidents could serve as a substrate for such collaborative learning while preserving privacy.

On the real-time communication front, future work could involve adding **biometric authentication** during the verification calls – for example, using voice recognition or gesture analysis to add additional automated checks on the user's legitimacy. As AI progresses, one could also imagine a more sophisticated AI agent handling the verification dialogue (an "AI fraud officer"), perhaps using Natural Language Processing to converse with the user and only involving a human staff if the conversation triggers certain risk cues. This could make the system more scalable across millions of users.

From the blockchain perspective, future research might examine the use of **smart contracts for automated remediation**. For instance, integrating with insurance or indemnity smart contracts that automatically reimburse customers for confirmed fraud, or linking with law enforcement networks for quick information sharing when fraud is confirmed (while respecting privacy). The performance of the blockchain under heavy load and its interoperability with existing bank ledgers is also worth exploring; technologies like sidechains or layer-2 solutions could be tested to ensure the audit trail remains scalable as transaction volumes grow.

Explainable AI in our system can be expanded by user-facing explainability – in future iterations, we could provide customers with simplified explanations when a transaction is flagged and challenged (e.g., "This

transfer is flagged because it's very different from your usual spending"). This might improve customer understanding and cooperation during the process. Ensuring that these explanations are easily comprehensible and do not inadvertently reveal system vulnerabilities (to avoid giving fraudsters hints) is a delicate balance for future user experience design.

Finally, a critical area for future work is **extensive real-world testing**. A pilot deployment in a controlled environment (like a sandbox with real customer transactions and opt-in users) would provide valuable feedback. Metrics from such a pilot would inform tweaks to the model thresholds, call workflows, and logging detail. It would also surface any unforeseen issues, such as certain demographics of users struggling with video verification (suggesting alternative verification modes), or operational challenges like handling a surge of verification calls during a coordinated fraud attack. Addressing these in subsequent iterations would move the concept closer to a production-ready solution.

In conclusion, **designing a real-time, AI-driven communication layer for fraud detection** as presented is a promising approach to bolster the security of financial services. It embodies a synergy of cutting-edge technologies – AI, real-time communications, and blockchain – to create a fraud prevention strategy that is proactive, transparent, and user-centric. As cybercriminals continue to evolve their tactics, such adaptive and multi-faceted defenses will be crucial in safeguarding digital banking platforms. We believe that the ideas and framework discussed in this work can inspire further innovation and eventually contribute to a safer financial ecosystem where customers can transact with confidence.

## References


[1] ABBASSI Hanae, BERKAOUI Abdellah, ELMENDILI Saida, & Youssef, G. (2023). End-To-End Real-Time Architecture For Fraud Detection In Online Digital Transactions. 14(6). Https://Doi.Org/10.14569/Ijacsa.2023.0140680

[2] Almaiah, M. A., & Maleh, Y. (2024). Machine Intelligence Applications In Cyber-Risk Management. IGI Global.

[3] Andrew Nii Anang, Oluwatosin Esther Ajewumi, Tobi Sonubi, Kenneth Chukwujekwu Nwafor, John Babatope Arogundade, & Itiade James Akinbi. (2024). Explainable AI In Financial Technologies: Balancing Innovation With Regulatory Compliance. International Journal Of Science And Research Archive, 13(1), 1793–1806. Https://Doi.Org/10.30574/Ijsra.2024.13.1.1870

[4] Arham, M. W. (2025). Transforming Auditing Through AI And Blockchain: A Comprehensive Study On Adoption, Implementation, And Impact In Financial Audits. American Journal Of Industrial And Business Management, 15(02), 225–241. Https://Doi.Org/10.4236/Ajibm.2025.152011

[5] Aros, L. H., Ximena, L., Gutierrez-Portela, F., Johver, J., & Samuel, M. (2024). Financial Fraud Detection Through The Application Of Machine Learning Techniques: A Literature Review. Humanities And Social Sciences Communications, 11(1). Https://Doi.Org/10.1057/S41599-024-03606-0

[6] Arun. (N.D.). Building Smarter Data Systems Leveraging Generative AI And Deep Learning. JEC PUBLICATION.

[7] Bello, A., & Olufemi, K. (2024). Artificial Intelligence In Fraud Prevention: Exploring Techniques And Applications Challenges And Opportunities. Computer Science & IT Research Journal, 5(6), 1505–1520. Https://Doi.Org/10.51594/Csitrj.V5i6.1252

[8] Brij Gupta, & Megha Quamara. (2020). Internet Of Things Security : Principles, Applications, Attacks, And Countermeasures. Crc Press, Taylor & Francis Group.

[9] Exploring The Benefits Of Blockchain Technology In Fraud Detection And Prevention. (2023, April 21). Finextra Research. Https://Www.Finextra.Com/Blogposting/24091/Exploring-The-Benefits-Of-Blockchain-Technology-In-Fraud-Detection-And-Prevention

[10] Flinders, M., Smalley, I., & Schneider, J. (2025, April 30). AI Fraud Detection In Banking. Ibm.Com. Https://Www.Ibm.Com/Think/Topics/Ai-Fraud-Detection-In-Banking

[11] IBM. (2024, December 4). Live Streaming. Ibm.Com. Https://Www.Ibm.Com/Think/Topics/Live-Streaming

[12] Ida, S. J., K. Balasubadra, R, S. R., & Lakshmi Narayanaa T. (2024). Enhancing Credit Card Fraud Detection Through LSTM-Based Sequential Analysis With Early Stopping. 3, 1–6. Https://Doi.Org/10.1109/Icnwc60771.2024.10537550

[13] Kaushik, K., Shubham Tayal, Susheela Dahiya, & Ayodeji Olalekan Salau. (2022). Sustainable And Advanced Applications Of Blockchain In Smart Computational Technologies. CRC Press.

[14] Lexisnexis® Risk Solutions. (2025). Lexisnexis.Com. Https://Globalsolutions.Risk.Lexisnexis.Com/Emailage-For-Adobe-Commerce?Trmid=BSCRMK25.DIGITAL.ECOM2.PHGO-1504531&Utm_Source=Google&Utm_Medium=Paid-Search&Utm_Campaign=BSCRMK25.DIGITAL.ECOM2&S_Kwcid=AL

[15] Mazumder, P. T. (2025). Blockchain In Trade Finance: Reducing Fraud And Improving Efficiency Through Digital Ledger Technology. Https://Doi.Org/10.2139/Ssrn.5255022

[16] Musleh Al-Sartawi, A. M. A. (2022). Artificial Intelligence For Sustainable Finance And Sustainable Technology : Proceedings Of ICGER 2021. Springer.

[17] Oladipupo Dopamu, Chika Okonkwo, Samuel Adeniji, & Femi Oke. (2024). Secure Messaging Application Using Java Cryptographic Architecture (JCA). World Journal Of Advanced Research And Reviews, 22(2), 2056–2063. Https://Doi.Org/10.30574/Wjarr.2024.22.2.1670

[18] Oladipupo Dopamu, Joseph Adesiyan, & Femi Oke. (2024). Artificial Intelligence And US Financial Institutions: Review Of AI-Assisted Regulatory Compliance For Cybersecurity. World Journal Of Advanced Research And Reviews, 21(3), 964–979. Https://Doi.Org/10.30574/Wjarr.2024.21.3.0791

[19] P, B. (2025, May 14). Video KYC In 2025: What It Is, How It Works & Why It Matters. Shuftipro.Com. Https://Shuftipro.Com/Blog/Video-Kyc-In-2025-What-It-Is-How-It-Works-Why-It-Matters/

[20] Pudi, M. (2024). REAL-TIME FRAUD DETECTION: A BANKING INDUSTRY CASE STUDY. International Journal Of Computer Engineering And Technology (IJCET), 15(6), 15–21. Https://Doi.Org/10.5281/Zenodo.14360088

[21] Putthiporn Thanathamathee, Siriporn Sawangarreerak, Siripinyo Chantamunee, & Mohd, N. (2024). SHAP-Instance Weighted And Anchor Explainable AI: Enhancing Xgboost For Financial Fraud Detection. Emerging Science Journal, 8(6), 2404–2430. Https://Doi.Org/10.28991/Esj-2024-08-06-016

[22] S. Suriya. (2025). Credit Card Fraud Detection Using Explainable AI Methods. Journal Of Information Systems Engineering And Management, 10(24s), 415–428. Https://Doi.Org/10.52783/Jisem.V10i24s.3917

[23] Tawakalit Ibiyeye, Iornenge, J. T., & Ayodeji Adegbite. (2024). Evaluating Regulatory Compliance In The Finance And Investment Sector: An Analysis Of Current Practices, Challenges, And The Impact Of Emerging Technologies. IOSR Journal Of Economics And Finance, 15(6), 1-08. Https://Doi.Org/10.9790/5933-1506010108
[24] Team, E. (2023, November 17). 2023 APP Fraud Trends And Changing Liability At A Glance. Finextra Research; Finextra. Https://Www.Finextra.Com/The-Long-Read/858/2023-App-Fraud-Trends-And-Changing-Liability-At-A-Glance
[25] The Intersection Of Artificial Intelligence And Cybersecurity: Safeguarding Data Privacy And Information Integrity In The Digital Age. (2024). International Journal Of Computer Applications Technology And Research. Https://Doi.Org/10.7753/Ijcatr1309.1002