Real-Time Mule Account Detection System for UPI Payments

Team Submission for Fintech Security Challenge
Problem Statement: Mule Accounts & Collusive Fraud in UPI

---

1. Understanding the Problem

1.1 The Growing Challenge

With the rapid growth of UPI in India, fraud techniques have also evolved. One of the most serious problems is the misuse of mule accounts, which are difficult to detect using traditional rule-based systems. From what we learned during our research, UPI processed over 130 billion transactions in 2024 alone. This massive scale makes it really hard to spot fraudulent patterns manually.

What are mule accounts?
These are bank accounts that fraudsters use as intermediaries to move stolen money. The account holders might be unknowing victims or willing participants who get paid a small commission. The money flows through multiple accounts quickly, making it tough to trace back to the criminals.

Why is this a real problem?

- Traditional fraud detection relies on fixed rules (like "flag transactions above ₹50,000"), but fraudsters adapt quickly
- By the time banks notice suspicious activity, the money has already moved through 5-10 accounts
- Innocent account holders sometimes get their accounts frozen by mistake, which frustrates legitimate users
- Banks spend a lot of time investigating false alarms

1.2 Our Research Findings

We studied several recent papers on fraud detection, particularly the MuleTrack system developed by NPCI researchers. Here's what we found:

Current Detection Methods:

1. **Rule-based systems** - Simple but easy to bypass
2. **Machine Learning models** - Better but struggle with evolving fraud patterns
3. **Graph-based approaches** - Show promise but need more work

The Gap We Identified:
Most existing solutions either focus on individual transactions OR on network patterns, but not both together in

real-time. Also, many research papers use simulated data, which doesn't capture the complexity of actual UPI fraud.

---

2. Our Proposed Solution

2.1 Core Idea

We're proposing a hybrid detection system that combines three techniques:

1. **Temporal Behavior Tracking** - Monitor how accounts behave over time
2. **Graph Network Analysis** - Map connections between accounts to spot fraud rings
3. **Device Intelligence** - Track device patterns to catch repeated offenders

Think of it like this: instead of just looking at individual suspicious transactions, we're building a complete picture of how money flows through networks of accounts and who's controlling them.

2.2 Technical Approach

Phase 1: Baseline Detection (Weeks 1-4)

What we'll build:

- Transaction profiling system that tracks weekly patterns for each account
- Simple rule engine based on existing research (velocity checks, amount thresholds)
- Basic API endpoint for real-time scoring

Technical details:

```
Input: Transaction metadata (amount, timestamp, sender/receiver IDs)
Processing:
  - Calculate weekly aggregates (total inflow, outflow, transaction count)
  - Compare against historical baselines
  - Flag accounts with sudden changes
Output: Risk score (0-100) and alert flag
```

Why start here?
We need a working baseline to compare against. This also helps us understand normal vs suspicious patterns in real data.

Phase 2: Graph-Based Detection (Weeks 5-10)

What we'll add:

- Build transaction graphs where accounts are nodes and payments are edges

- Implement community detection algorithms to find tightly connected groups
- Use Graph Neural Networks (inspired by recent research papers we studied)

Our approach: After reading papers on GNN-based fraud detection, we realized that money mule rings have specific graph patterns:

- **Star pattern**: One central account receives from many, sends to one
- **Chain pattern**: Money flows through 5-10 accounts sequentially
- **Circular pattern**: Money loops back to origin after several hops

We'll train a GNN model to recognize these patterns. The model learns to look at not just what an account does, but who it interacts with.

Technical stack:

- Python with NetworkX for graph construction
- PyTorch Geometric for GNN implementation
- Redis for caching hot transaction paths
- PostgreSQL for storing graph structures

Phase 3: Device Fingerprinting (Weeks 11-14)

The insight: Fraudsters often use the same device to control multiple mule accounts. If we can link devices to accounts, we can catch entire fraud operations.

What we'll track:

- Device ID (from UPI app)
- IP addresses and their geographic patterns
- Login time patterns (fraudsters often operate at odd hours)
- App version and OS details

How it works:

1. Create a unique fingerprint for each device based on multiple attributes
2. Track which accounts access from the same device
3. If multiple flagged accounts share a device, increase risk scores for all connected accounts

Privacy considerations: We'll only store hashed device IDs and won't collect any personal data beyond what's already in UPI transaction logs.

Phase 4: Real-Time Integration (Weeks 15-16)

The challenge: UPI processes transactions in under 2 seconds. Our detection system needs to score transactions even faster without slowing down payments.

Our strategy:

- Pre-compute risk scores for known accounts (updated hourly)
- Use a lightweight scoring API for real-time transactions
- Only run heavy graph analysis for high-risk cases
- Implement a queue system for deeper investigation of flagged accounts

API Design:

```
POST /api/v1/score-transaction
Request: {
  "transaction_id": "TXN123456",
  "sender_vpa": "user@bank",
  "receiver_vpa": "merchant@bank",
  "amount": 5000,
  "timestamp": "2025-01-15T10:30:00Z"
}

Response: {
  "risk_score": 67,
  "risk_level": "MEDIUM",
  "flags": ["velocity_spike", "new_beneficiary"],
  "processing_time_ms": 45
}
```

2.3 Innovation in Our Approach

What makes our solution different:

1. **Hybrid scoring:** We combine multiple signals instead of relying on just one technique
2. **Adaptive thresholds:** Risk thresholds adjust based on account history (new accounts vs 5-year-old accounts have different baselines)
3. **Explainability:** When we flag an account, we provide specific reasons (unlike black-box ML models)
4. **Minimal false positives:** By using multiple confirmation signals, we reduce chances of blocking legitimate users

Learning from existing research: The MuleTrack paper showed that simple Markov Chain models can outperform complex deep learning in precision. We're taking that lesson seriously - sometimes simpler is better,

especially when you need to explain decisions to bank investigators.

---

3. System Architecture

3.1 Overall Design

```
[UPI Transaction Stream]
        ↓
[Real-Time Ingestion Layer]
  - Kafka topics for transaction events
  - 3-5 second processing SLA
        ↓
[Feature Engineering Pipeline]
  - Compute behavioral features
  - Extract graph features
  - Enrich with device data
        ↓
[Multi-Model Scoring Engine]
  - Temporal model (Markov Chain)
  - Graph model (GNN)
  - Device correlation model
        ↓
[Risk Aggregation & Decision]
  - Weighted ensemble scoring
  - Threshold-based alerting
        ↓
[Investigator Dashboard]
  - Flagged account clusters
  - Visual network graphs
  - Drill-down capabilities
```

3.2 Technology Choices

Why we chose these tools:

Component	Technology	Reason
Data Pipeline	Apache Kafka	Handles high throughput, proven in fintech
Graph Database	Neo4j	Native graph queries, good visualization
ML Framework	PyTorch	Flexible for custom GNN implementations
API Layer	FastAPI	Fast, async support, automatic docs
Monitoring	Prometheus + Grafana	Industry standard, easy setup
Cache	Redis	Sub-millisecond lookups for hot data

Infrastructure:

- Containerized deployment using Docker
- Kubernetes for orchestration (scales with transaction load)
- Separate processing pipelines for real-time vs batch analysis

---

4. Expected Outcomes

4.1 Performance Targets

Based on our study of existing research, here's what we aim to achieve:

Detection Metrics:

- **Precision:** 0.65-0.70 (similar to MuleTrack's 0.67)
- **Recall:** 0.55-0.60 (balance between catching fraud and false alarms)
- **F1-Score:** 0.60-0.65
- **False Positive Rate:** < 0.5% (crucial for user experience)

Operational Metrics:

- **Processing Latency:** < 100ms for real-time scoring
- **Throughput:** 10,000 transactions/second
- **Detection Time:** Flag suspicious accounts within 24 hours of first mule activity
- **Investigation Time:** Reduce analyst time by 40% with better clustering

4.2 Real-World Impact

For Banks:

- Faster identification of compromised accounts
- Reduced losses from money laundering
- Less time spent on false positive investigations

For Users:

- Fewer legitimate transactions getting blocked
- Better protection from account takeover
- Faster resolution when fraud does occur

For Regulators:

- Better compliance with AML requirements
- Data-driven insights into emerging fraud patterns
- Audit trail for every flagged transaction

4.3 Success Criteria

We'll know our system is working if:

1. We catch at least 60% of known mule accounts in test data
2. Less than 1 in 200 legitimate transactions get falsely flagged
3. The system can process peak UPI loads (New Year's Eve, festival sales)
4. Bank investigators can understand why accounts were flagged

---

5. Benchmarking Against Existing Solutions

5.1 Comparison Matrix

Feature	Traditional Rules	ML Models (RF/XGBoost)	Our Solution
Temporal Patterns	❌ Static	⚠️ Limited	✅ Full tracking
Network Analysis	❌ None	❌ None	✅ GNN-based
Device Correlation	❌ None	❌ None	✅ Fingerprinting
Real-time Scoring	✅ Fast	⚠️ Moderate	✅ Fast + Accurate

Feature	Traditional Rules	ML Models (RF/XGBoost)	Our Solution
Explainability	✅ Clear	❌ Black box	✅ Clear
Adaptation	❌ Manual	⚠️ Periodic retraining	✅ Continuous learning

5.2 Learning from Research

What we adopted from MuleTrack:

- Temporal state modeling using weekly windows
- Focus on precision over recall for operational feasibility
- Interpretable models for regulatory compliance

What we added beyond existing research:

- Integration of graph features with temporal features
- Device-level fraud correlation
- Real-time scoring API architecture
- Visual investigation tools

Limitations we acknowledge:

- Need significant labeled data for training (will use synthetic data initially)
- Graph analysis is computationally expensive for large networks
- Device fingerprinting can be bypassed by sophisticated fraudsters

---

6. Differentiation & Unique Value

6.1 What Makes Us Different

1. Multi-Signal Approach
Most fraud detection systems use either behavioral OR network analysis. We use both together, which catches more types of fraud:

- Behavioral model catches individual mule accounts
- Graph model catches organized fraud rings
- Device model catches the same criminals operating multiple accounts

2. Explainable AI
Banks need to explain to customers why their transaction was blocked. Our system provides clear reasons:

- "Account shows sudden spike in incoming transfers (5x normal)"
- "Account is connected to 3 other flagged accounts"
- "Device accessing this account has controlled 8 other suspicious accounts"

3. Built for India's Scale
UPI isn't just big - it's massive. We designed every component with scale in mind:

- Horizontal scaling for graph processing
- Caching strategies for hot accounts
- Batch + real-time processing separation

4. Practical Deployment Focus
Many research papers stop at accuracy numbers. We're thinking about:

- How do banks integrate this into their systems?
- What happens when fraudsters adapt?
- How do we update models without downtime?

6.2 Innovation Beyond Research

While we learned a lot from academic papers, we're adding practical innovations:

Smart Thresholding: Instead of fixed thresholds, we use account-age-adjusted thresholds:

- New accounts (< 3 months): Lower threshold for flagging
- Established accounts (> 2 years): Higher threshold to avoid false positives

Fraud Pattern Library: As we detect new fraud patterns, we automatically add them to a pattern library that other models can learn from.

Collaborative Filtering: If one bank flags an account, and that account interacts with accounts at other banks, we can warn those banks (with privacy protections).

---

7. Product Roadmap & Use Cases

7.1 Development Roadmap

Phase 1 (Month 1-2): MVP

- [ ] Basic transaction monitoring API

- [ ] Rule-based flagging system
- [ ] Simple dashboard for viewing flagged accounts
- [ ] Integration with sample UPI data

Phase 2 (Month 3-4): Enhanced Detection

- [ ] Deploy Markov Chain temporal model
- [ ] Add graph construction pipeline
- [ ] Implement device fingerprinting
- [ ] Build investigator interface

Phase 3 (Month 5-6): Graph Intelligence

- [ ] Train and deploy GNN model
- [ ] Community detection algorithms
- [ ] Network visualization tools
- [ ] Advanced risk scoring

Phase 4 (Month 7-8): Production Ready

- [ ] Performance optimization
- [ ] Scalability testing
- [ ] Security hardening
- [ ] Bank integration APIs
- [ ] Documentation and training

7.2 Real-World Use Cases

Use Case 1: Student Falls Victim to Job Scam *Scenario:* A student gets a "work from home" offer. They're asked to receive money in their account and transfer it elsewhere, keeping 2% commission.

How our system helps:

1. After the first few transactions, behavioral model flags unusual pattern
2. Graph analysis shows the student's account is part of a larger network
3. Bank gets alert within 24 hours, freezes account
4. Student is saved from becoming an unwitting mule

Use Case 2: Organized Fraud Ring *Scenario:* Criminals create 50 mule accounts using fake/stolen documents. They cycle money through these accounts.

How our system helps:

1. Graph model identifies the tightly connected cluster
2. Device fingerprinting shows many accounts accessed from same devices

3. System flags the entire network, not just individual accounts
4. Law enforcement can investigate the whole operation

Use Case 3: Compromised Account *Scenario:* A legitimate user's phone is stolen. Fraudster uses it to receive stolen money.

How our system helps:

1. Behavioral model detects sudden change in transaction patterns
2. Device fingerprint shows access from new device
3. Combined signals trigger high-risk alert
4. Bank contacts user to verify before processing large transfers

Use Case 4: Merchant Account Abuse *Scenario:* Fraudsters convert personal accounts to merchant accounts and create fake transactions.

How our system helps:

1. Temporal model catches the sudden shift in transaction types
2. Graph analysis shows unusual patterns (same customers repeatedly)
3. System flags for manual review before significant damage

7.3 Go-to-Market Strategy

Target Users:

1. **Primary:** Banks and Payment Service Providers (PSPs) in India
2. **Secondary:** Fintech companies using UPI
3. **Tertiary:** Regulatory bodies for compliance monitoring

Deployment Options:

- **Cloud SaaS:** Banks can integrate via API
- **On-Premise:** For banks with strict data residency requirements
- **Hybrid:** Real-time scoring in cloud, sensitive data on-premise

Pricing Model (Hypothetical):

- Per-transaction fee: ₹0.01 per transaction scored
- Monthly license: ₹5,00,000 for up to 10M transactions
- Enterprise: Custom pricing with dedicated support

8. Technical Challenges We Anticipate

8.1 Challenges and Our Solutions

Challenge 1: Data Imbalance

- *Problem:* Fraud is rare (< 0.1% of transactions)
- *Our solution:* Use SMOTE for synthetic fraud samples during training, focus on precision rather than overall accuracy

Challenge 2: Concept Drift

- *Problem:* Fraudsters change tactics every few months
- *Our solution:* Continuous monitoring of model performance, automated retraining triggers when accuracy drops

Challenge 3: Cold Start Problem

- *Problem:* New accounts have no history
- *Our solution:* Use stricter initial thresholds, gradually relax as account builds history

Challenge 4: Privacy Concerns

- *Problem:* Can't share data across banks
- *Our solution:* Use federated learning techniques, only share aggregate patterns not individual transactions

Challenge 5: Real-Time Performance

- *Problem:* Graph analysis is slow
- *Our solution:* Pre-compute risk scores for known accounts, only do full graph analysis for flagged cases

8.2 What Could Go Wrong

Being honest about potential failures:

1. **Model Bias:** If training data has biases, we might unfairly flag certain user groups
    - *Mitigation:* Regular fairness audits, diverse training data
2. **False Positives:** Legitimate businesses might get flagged
    - *Mitigation:* Whitelist mechanism, human review for high-value accounts
3. **Scalability Issues:** System might not handle festival season loads

- *Mitigation:* Extensive load testing, auto-scaling infrastructure

4. **Adversarial Attacks:** Fraudsters might figure out our detection logic
    - *Mitigation:* Regular model updates, ensemble approach makes it harder to game

---

9. Conclusion

9.1 Summary

We've proposed a practical, scalable solution for detecting mule accounts in UPI that combines:

- Temporal behavior tracking (learning from MuleTrack research)
- Graph neural networks (inspired by recent GNN fraud detection papers)
- Device intelligence (adding a new dimension to detection)

Our solution is designed to be:

- **Fast enough** for real-time UPI transactions
- **Accurate enough** to minimize false positives
- **Explainable enough** for bank investigators and regulators
- **Scalable enough** for India's massive payment infrastructure

9.2 Why This Matters

UPI has become critical infrastructure for India. When fraud goes undetected, it doesn't just hurt banks - it hurts small merchants, gig workers, and regular people who lose their hard-earned money. By catching mule accounts faster, we can:

- Stop fraud before money leaves the country
- Protect innocent people from becoming unwitting mules
- Make digital payments safer for everyone

9.3 Next Steps

If selected for this challenge, we would:

1. **Week 1-2:** Set up development environment, acquire sample data
2. **Week 3-4:** Build baseline detection system
3. **Week 5-8:** Implement graph analysis and device fingerprinting
4. **Week 9-12:** Testing, optimization, and dashboard development

5. **Week 13-16:** Documentation, demo preparation, final presentation

We're excited about this problem because it's not just a technical challenge - it's a chance to make India's digital payment system more secure for millions of users.

References

1. Jambhrunkar, G., et al. (2025). "MuleTrack: A Lightweight Temporal Learning Framework for Money Mule Detection in Digital Payments." IWANN 2025.
2. Cheng, D., et al. (2024). "Graph Neural Networks for Financial Fraud Detection: A Review." arXiv:2411.05815.
3. Ghosh, S., et al. (2023). "Anti-Money Laundering by Group-Aware Deep Graph Learning." IEEE Transactions on Knowledge and Data Engineering.
4. Johannessen, F., & Jullum, M. (2023). "Finding Money Launderers Using Heterogeneous Graph Neural Networks." arXiv:2307.13499.
5. NPCI Annual Report 2024. "UPI Transaction Statistics."

Document prepared by: [Your Team Name]
Date: December 2025
Contact: [Your Email]

Note: This submission is entirely our original work based on our research and understanding of the problem. All technical approaches are our own proposals based on studying existing research papers.