



Krishna Sankar
@ksankar

Graph For A Better Token Economy

- **Problem Statement**

- Immutable, Distributed, Unanimous & *Secure* DLT & Blockchain stack for Cryptocurrency & Web 3.0
 - While the other properties are well implemented, security is still a challenge
- Specifically, this solution addresses the detection of Fraud Rings (e.g., Money Laundering) & Camouflaged Fraud (e.g., Fake Product Reviews)

- **Hypothesis**

- Graph representation & algorithms are well suited to extract fraud rings of arbitrary complexity

- **Approach**

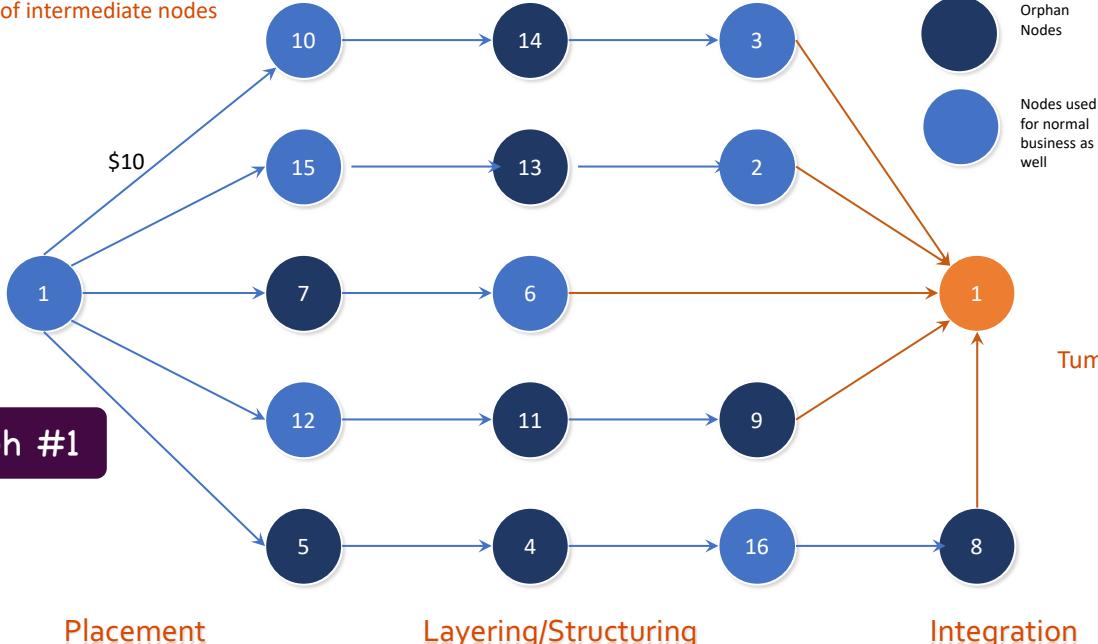
- Tackle one of the complex fraud pattern in a scalable, extensible way

Fan-In/Fan-Out Fraud Rings

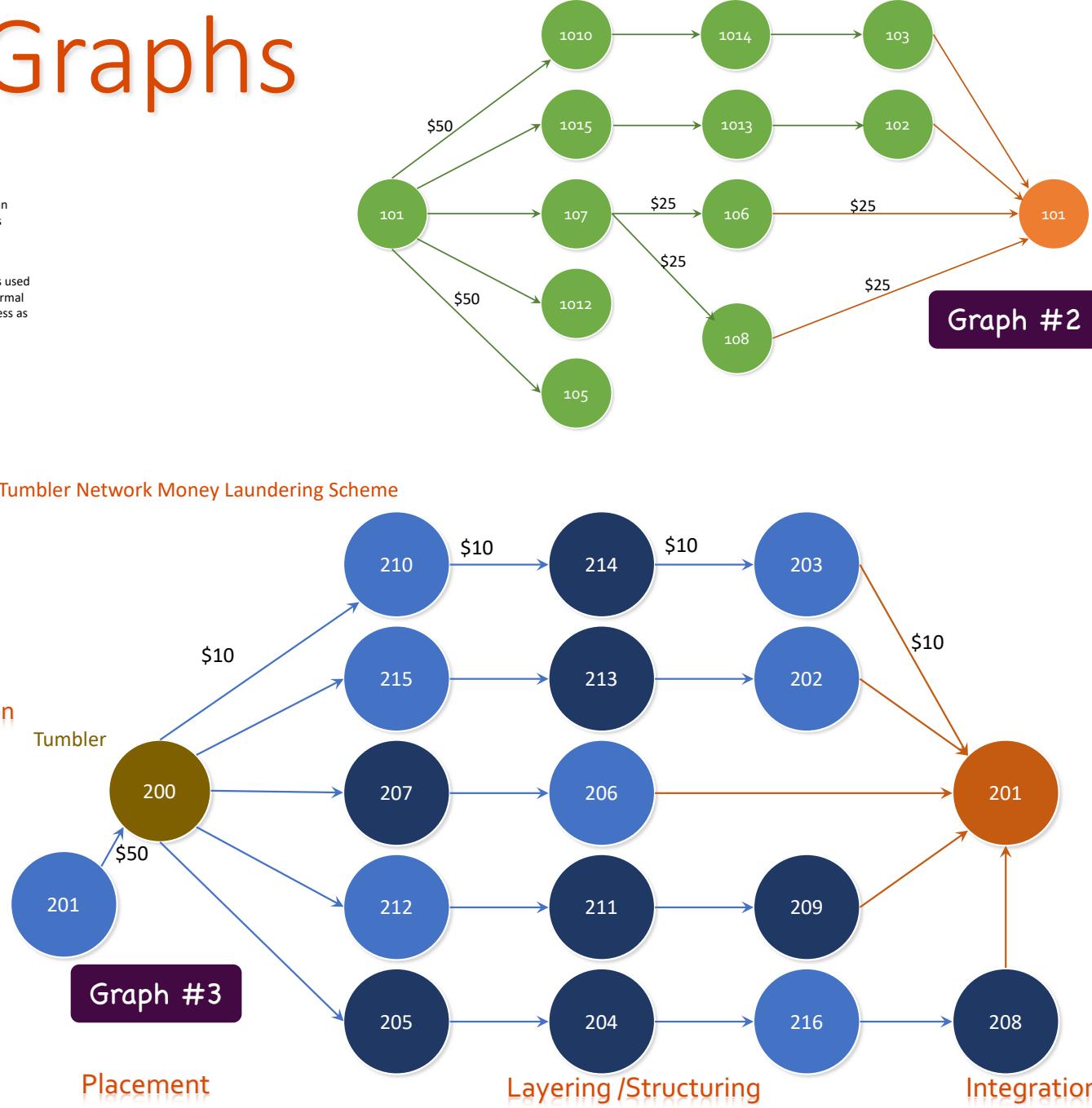
- Money Laundering goes through 3 stages viz Placement, Layering and Integration
- Layering is the most complex, where the money is piped through a complex web of strategic transactions – mixers, tumblers, long chain of crypto transactions et al., obscuring the money trail
- As we will see later, one essential aspect in understanding the problem is to draw representative graphs – the happy path 1st (see next slide)

Money Laundering Graphs

Simple Money Laundering Scheme
with lots of intermediate nodes



Tumbler Network Money Laundering Scheme



Challenges

- Payment network is very different from social or other networks
 - Temporal, monotonic time
 - Not symmetric
 - Other attributes (like amount) matters
 - Many of the concepts like connected components and page rank do not mean much
- Detecting fraud ring is literally finding the proverbial needle in a haystack
 - Class imbalance, Long Fraud Chains

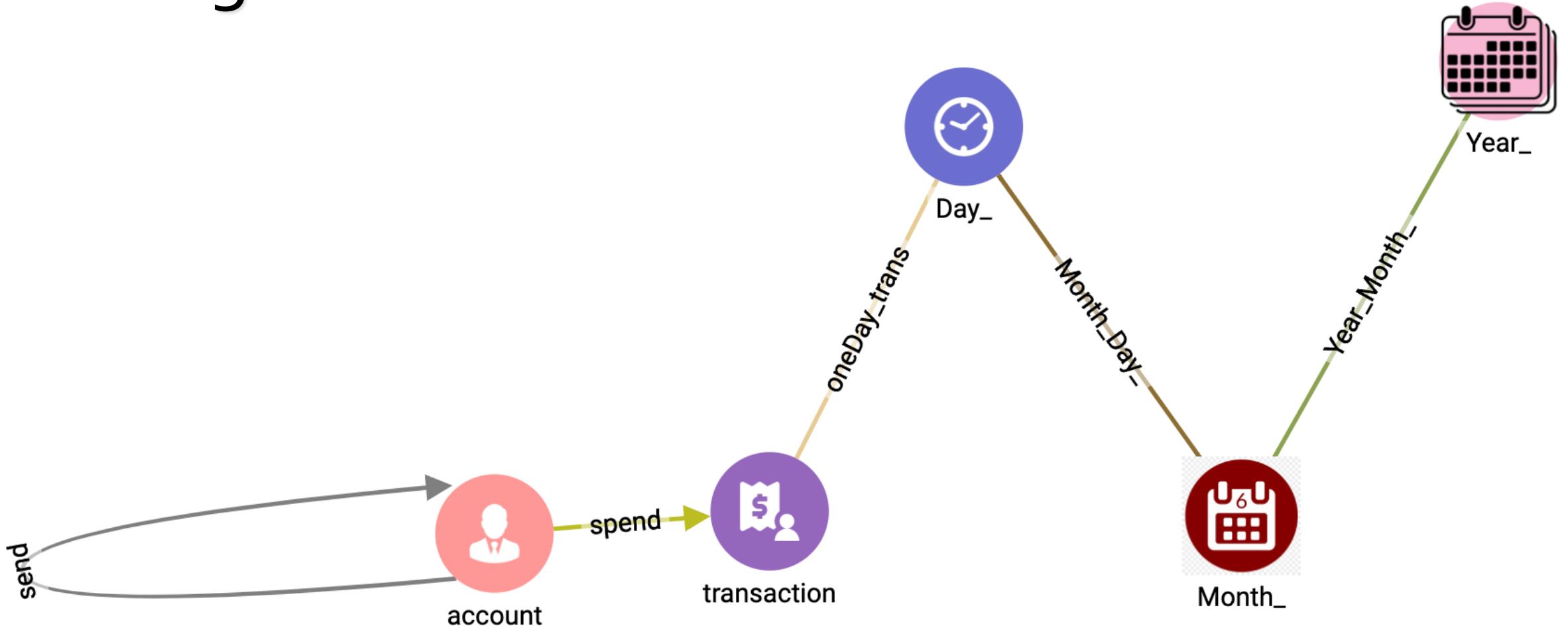
Solutions

- Layered approach with well defined pipeline stages
 - *Overlay fraud rings progressively*
 - *Narrow down and organize the vertices as we progress*
 - *Always keep time sequencing in the processing*
- Customize relevant graph algorithms
 - *e.g., Rocha-Thatte Cycle Detection, but it is unordered*
- Add runtime attributes to vertices
 - *That will help the processing downstream the pipeline*
- Start with a simple schema and add more elements as required
- Stay in TigerGraph as much as possible



Demo

Design Schema

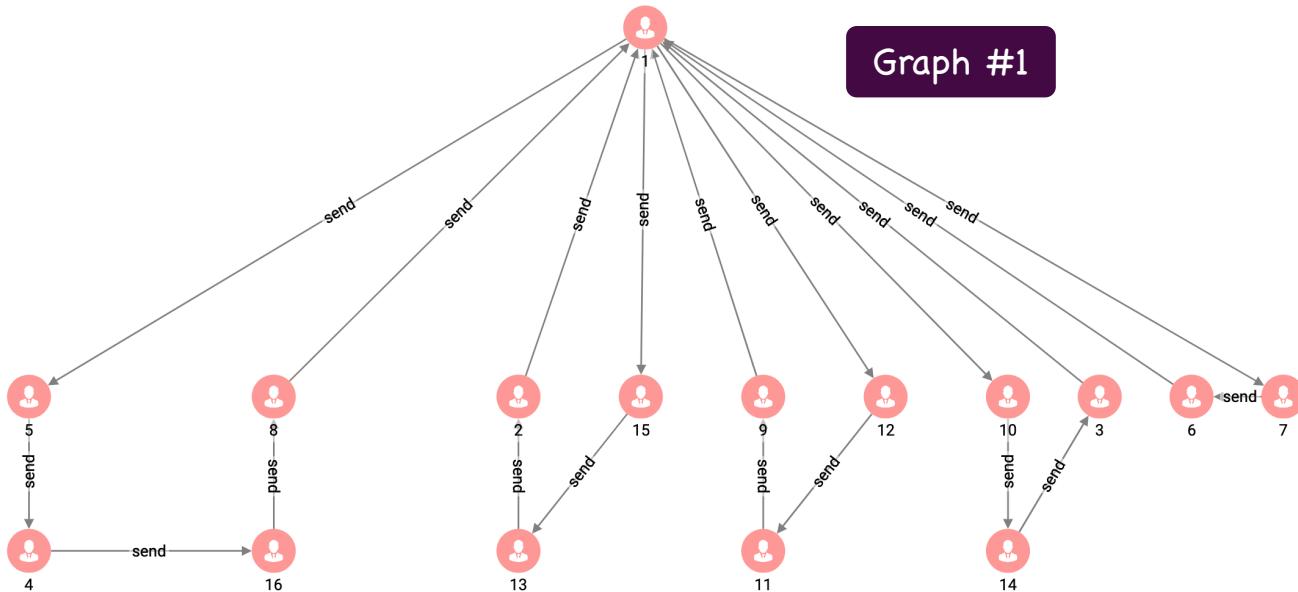


Schema Information

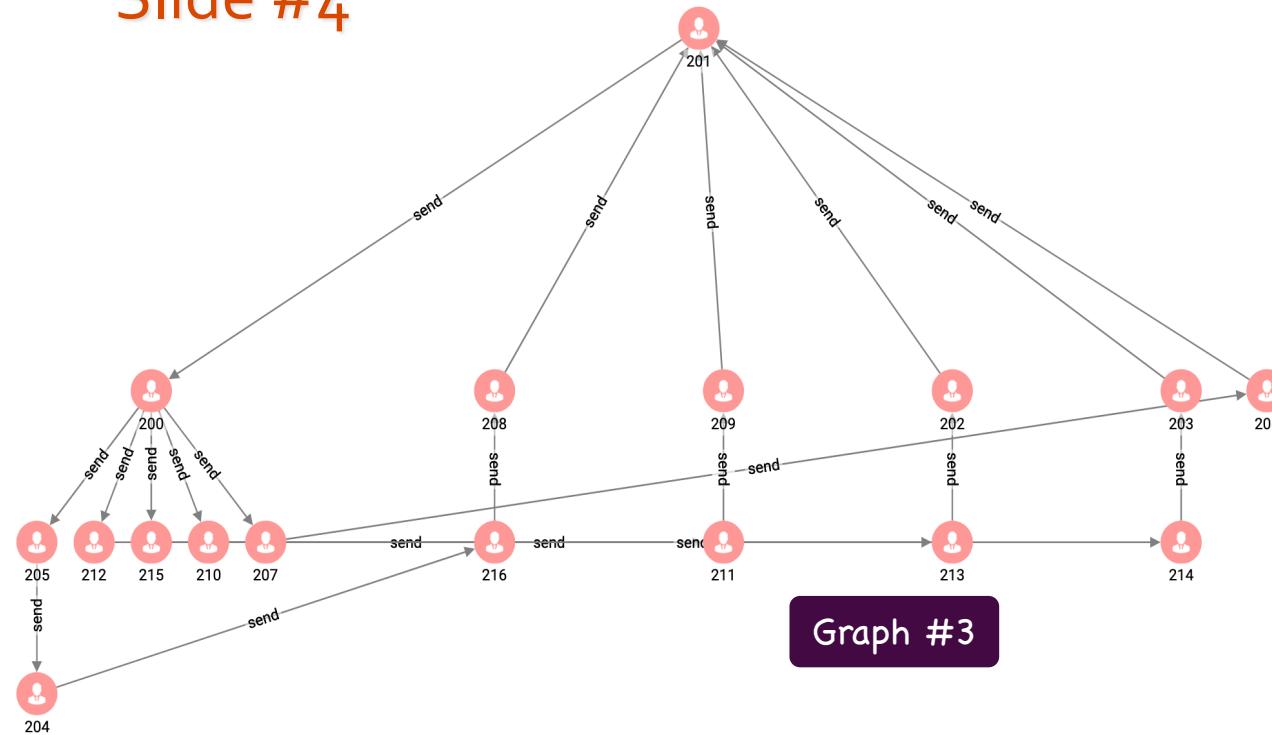
- Vertex information:
 - Vertex: account
 - (PRIMARY ID) id: STRING
 - v_type: INT
 - Vertex: transaction
 - (PRIMARY ID) id: STRING
 - amt: DOUBLE
 - date_time: DATETIME
 - from_id: UINT
 - to_id: UINT
 - Vertex: Year_
 - (PRIMARY ID) id: STRING
 - Vertex: Month_
 - (PRIMARY ID) id: STRING
 - Vertex: Day_
 - (PRIMARY ID) id: STRING
- Edge information:
 - Edge: send
 - Source: account
 - Target: account
 - amount: DOUBLE
 - date_time: DATETIME
 - reverse edge: reverse_send
 - Edge: spend
 - Source: account
 - Target: transaction
 - reverse edge: reverse_spend
 - Edge: Year_Month_
 - Source: Year_
 - Target: Month_
 - Edge: Month_Day_
 - Source: Month_
 - Target: Day_
 - Edge: oneDay_trans
 - Source: transaction
 - Target: Day_
 - date_time: DATETIME

Output

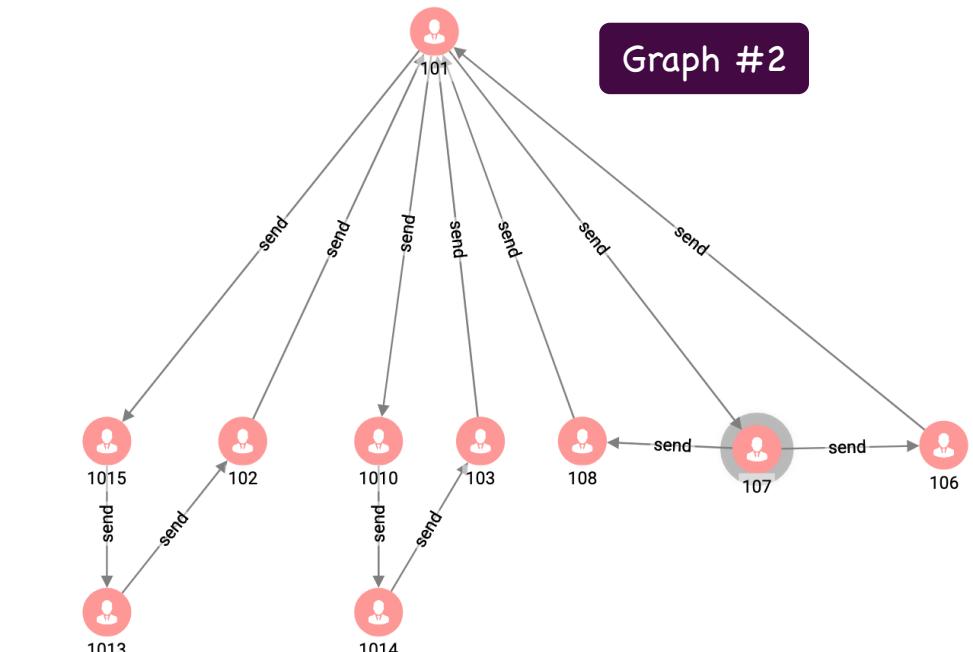
- It finds all the 3 fraud rings !
- And, correctly identifies the fraud actors at the helm
- You can compare with Slide #4



Graph #1



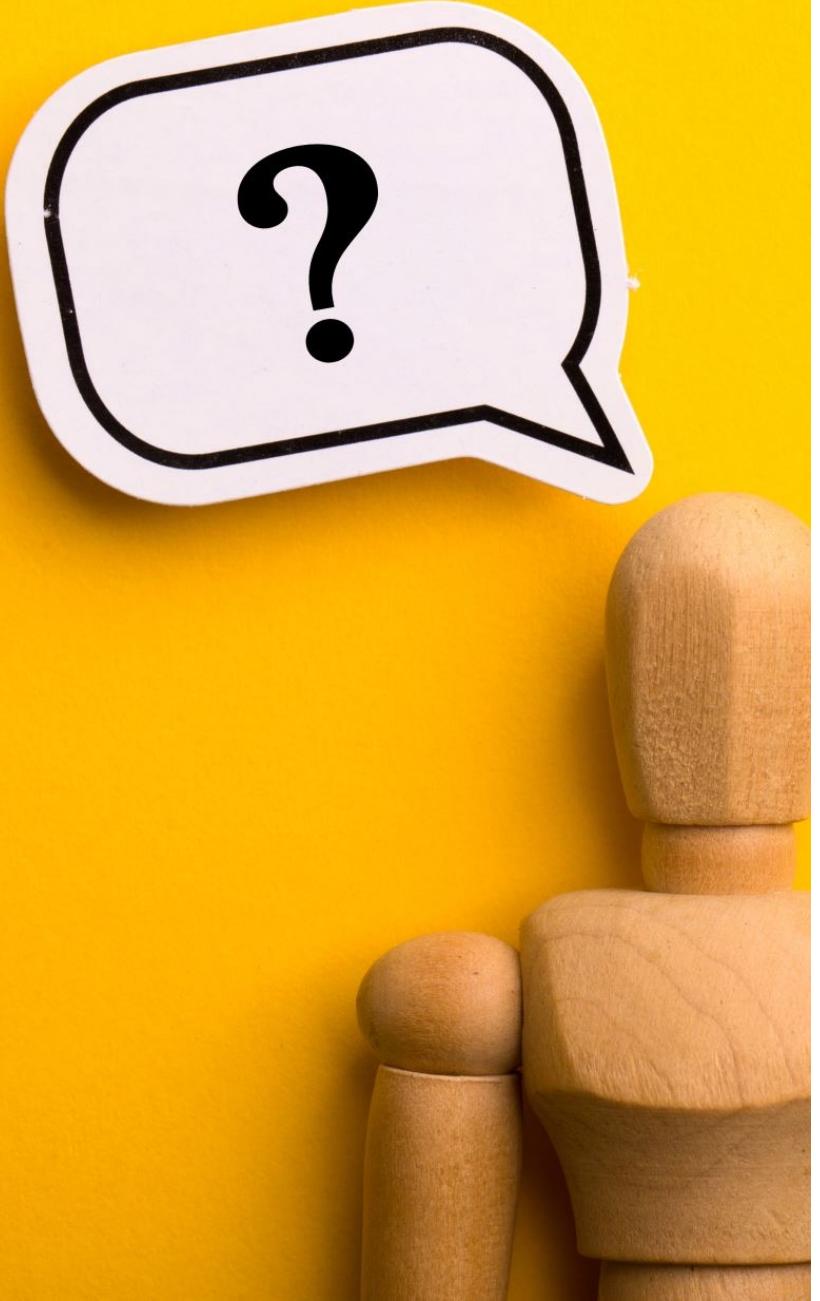
Graph #3



Graph #2

Lessons Learned

1. Graph representation is appropriate for DLT/Blockchain Fraud Detection
 - *TigerGraph is a feature rich, flexible & scalable Graph Database well suited for this problem*
 - *But it requires disciplined thinking, at times different than what we are used to !*
2. Spend time thinking about & understanding the problem
3. Make simplified assumptions and relax them as you progress (Layered approach)
4. Think Graphs & more specifically Parallel Graphs – It will take a little time to get used to that concept
 - *Read, write & learn the patterns from the Graph Algorithms GSQL code*
5. Draw graph diagrams to visualize the problem - *Draw the happy path 1st & then edge cases*
6. Create datasets depicting multiple scenarios – *1st use a small dataset to test the algorithms*
7. Do as much in TigerGraph as possible
 - *It is tempting to process a list outside (say in python), after a quick GSQL algorithm; don't stop there, persist (using GSQL) until you have exhausted all graph ideas*



Things To Try Next

This is only the beginning !!

1. Fine grained temporals
 - There could be many such rings by the same actors, so need to separate the rings by time
 - solution : time tree “If you need to filter use vertices” – TigerGraph pragma
 - Opportunity for Payment Networks in TigerGraph Graph Algorithms
2. More expressive Graph schema with derived runtime attributes, especially to track cross-ledger behaviors
3. Explore Graph Motif extraction
4. Weighted Graphs
5. Cross-Ledger tracking of fraud rings
6. Graph Neural Networks leveraging the extended dynamic attributes
7. Entity Resolution
 - Need to understand heavy spans & differentiate between Exchanges, Tumblers, Mixers et al
 - Probably highest measure of eigenvalue centrality
 - Add Vertex type based customized logic