Detecting Illicit Behavior in Crypto Transaction Networks

Project Proposal,
Howard Wang
Yazhe Huang

Table of contents

Mission statement & Research motivation

According to PeckShield's 2024 Crypto Security Report, crypto-related scams surged with $3.01B in stolen assets—up *15%* from 2023. As crypto becomes part of everyday life, security improvements are essential. Our study uses graph theory and network analysis to uncover patterns in fraudulent behavior, aiming to better detect and prevent similar incidents in the future.

Project Progress - Dataset

```
folder_path = "/content/drive/My Drive/"

file_path_1 = folder_path + "first_order_df.csv"
file_path_2 = folder_path + "transaction_dataset.csv"
file_path_3 = folder_path + "second_order_df.csv"
file_path_4 = folder_path + "addresses.csv"

import pandas as pd

df_first_order = pd.read_csv(file_path_1)
df_transaction = pd.read_csv(file_path_2)
df_second_order = pd.read_csv(file_path_3)
df_addresses = pd.read_csv(file_path_4)

print(df_first_order.info())
print(df_transaction.info())
print(df_second_order.info())
print(df_addresses.info())
```

Imported and used Kaggle Dataset, ethereum-transactions-for-Fraud-detection.

Project Progress - Dataset

Constructed the transaction graph and calculated degree centrality to identify key wallet addresses.

```
df_small = df_combined.head(10000)

G = nx.DiGraph()

for index, row in df_small.iterrows():
    sender = row["From"]
    receiver = row["To"]
    value = row["Value"]

    G.add_edge(sender, receiver, weight=value)

print(f"Graph created with {G.number_of_nodes()} nodes and {G.number_of_edges()} edges.")
```

```
Graph created with 5689 nodes and 6553 edges.
```

```
[10] degree_centrality = nx.degree_centrality(G)

df_degree_centrality = pd.DataFrame({
    "Address": list(degree_centrality.keys()),
    "Degree Centrality": list(degree_centrality.values())
}).sort_values(by="Degree Centrality", ascending=False)

df_degree_centrality.head()
```

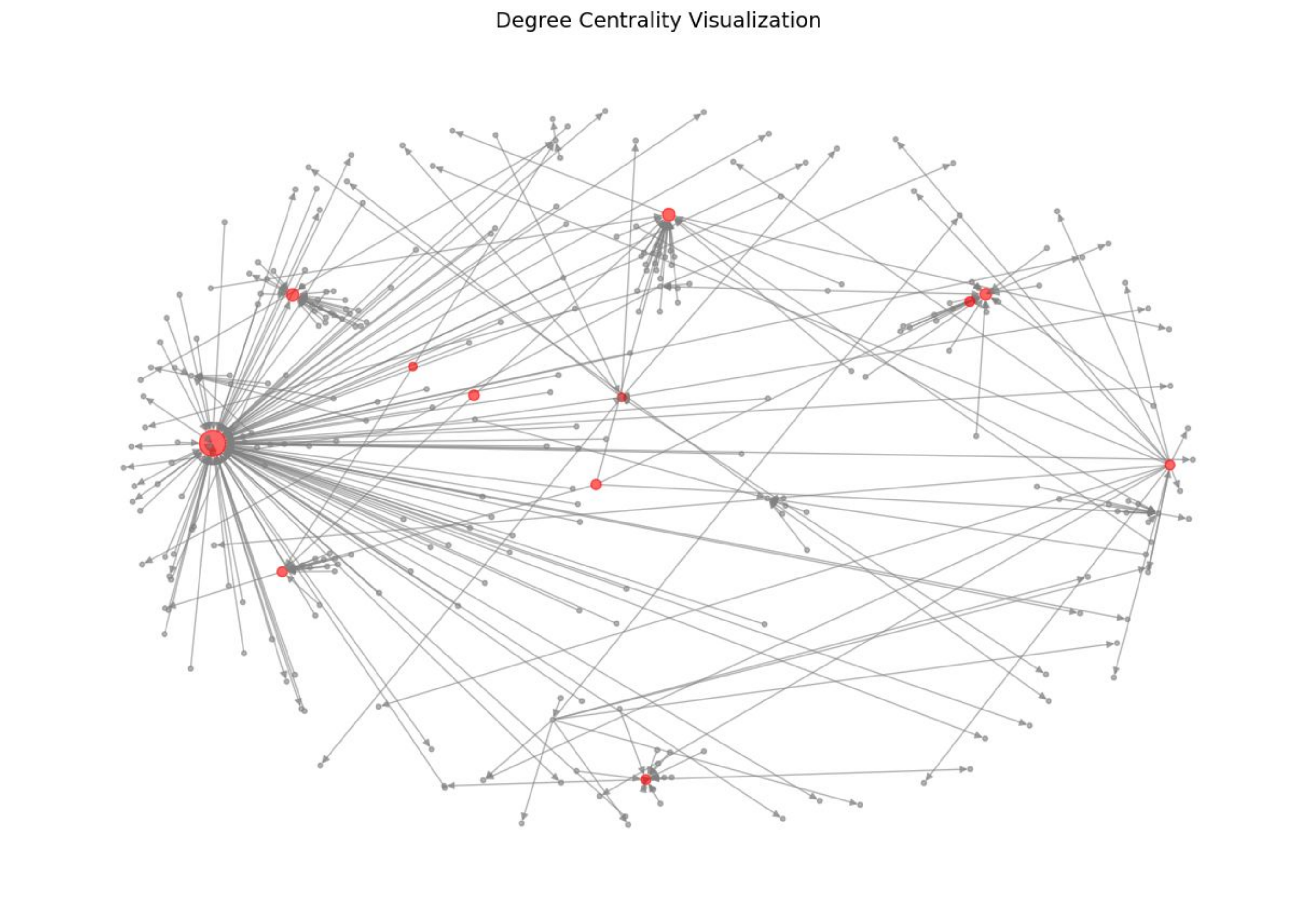
	Address	Degree Centrality
730	0x062263af47ffa76f1b4b5c3aa0ef2b62ade436ea	0.086472
1172	0x0ae775637e63fa95855246fd82e96802d05883fc	0.074801
419	0x0591a42188996397fc7cd6db729045146c37696c	0.062069
6	0x0059b14e35dab1b4eee1e2926c7a5660da66f747	0.061538
1788	0x0fa3202e9f247f9349352c4296d2bbd6c7fa11b4	0.061008

Project Progress - Active Wallets

Visualized the top 50 active wallets within a subgraph to highlight their central roles in the transaction network.

```
import matplotlib.pyplot as plt

# Extract top 50 most active wallets
top_addresses = df_degree_centrality["Address"].head(50).tolist()

# Get subgraph of first 300 nodes
subgraph_nodes = list(G.nodes())[:300]
subgraph = G.subgraph(subgraph_nodes)

# Compute positions
pos = nx.spring_layout(subgraph, seed=42)

# **Fix node_size and node_color to match subgraph**
node_size = [degree_centrality[node] * 5000 if node in top_addresses else 10 for node in subgraph.nodes()]
node_color = ["red" if node in top_addresses else "gray" for node in subgraph.nodes()]

# Plot graph
plt.figure(figsize=(12, 8))
nx.draw(subgraph, pos, node_size=node_size, node_color=node_color, edge_color="gray", alpha=0.6, with_labels=False)

plt.title("Degree Centrality Visualization", fontsize=14)
plt.axis("off")
plt.show()
```

Result



Project Progress - PageRank

```
# Compute PageRank
pagerank_scores = nx.pagerank(G, alpha=0.85)

# Convert to DataFrame
df_pagerank = pd.DataFrame({
    "Address": list(pagerank_scores.keys()),
    "PageRank Score": list(pagerank_scores.values())
}).sort_values(by="PageRank Score", ascending=False)

# Show Top 10 Influential Wallets
print(df_pagerank.head(10))
```

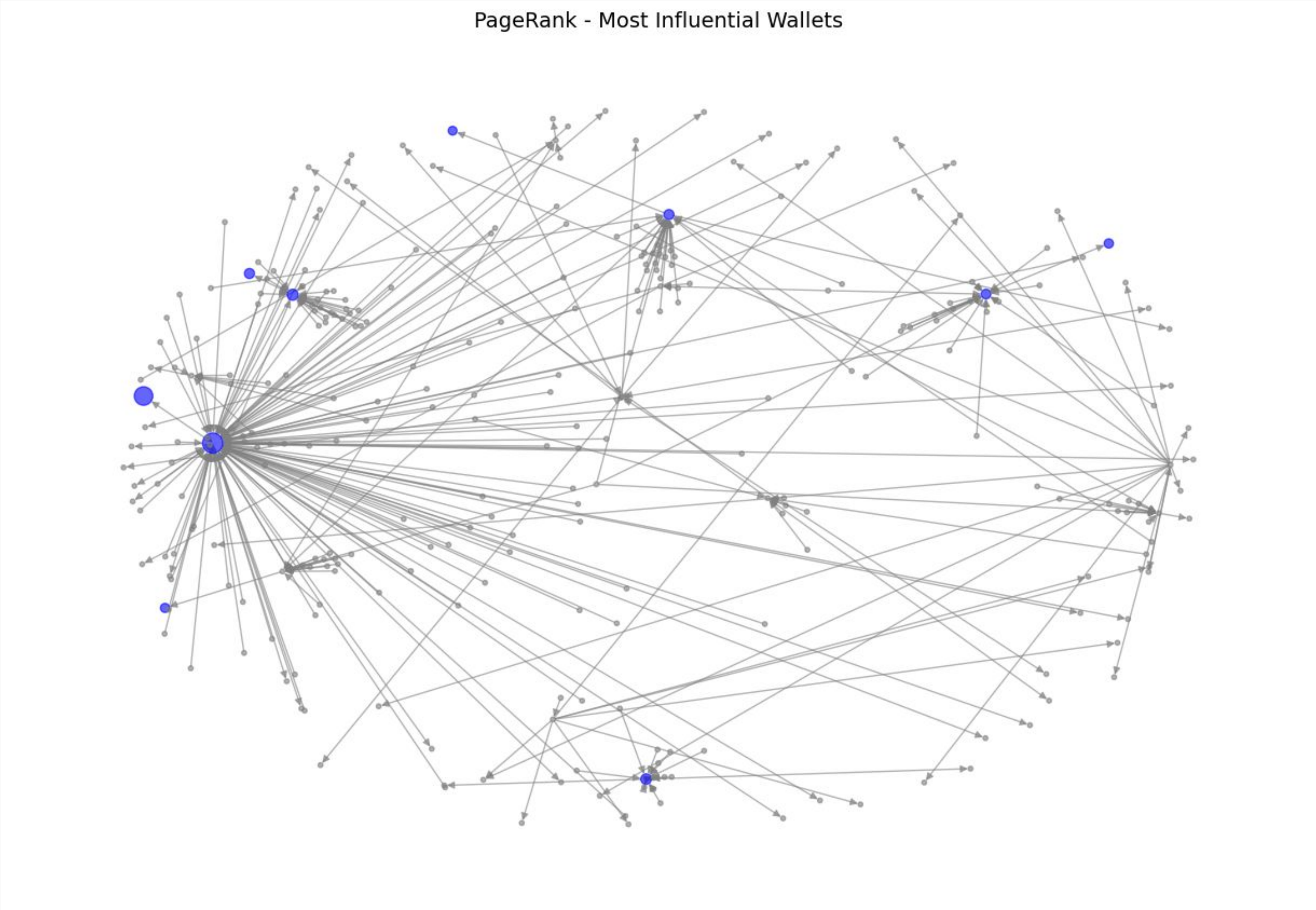
Computed PageRank scores to identify the top influential wallets based on their connectivity in the transaction graph.

Project Progress - PageRank

Visualized top PageRank wallets to highlight the most influential addresses in the transaction network.

```
# Extract top 50 most influential wallets
top_pagerank_addresses = df_pagerank["Address"].head(50).tolist()

# Get subgraph of first 300 nodes
subgraph_nodes = list(G.nodes())[:300]
subgraph = G.subgraph(subgraph_nodes)

# Compute positions
pos = nx.spring_layout(subgraph, seed=42)

# Fix node_size & node_color
node_size = [pagerank_scores[node] * 10000 if node in top_pagerank_addresses else 10 for node in subgraph.nodes()]
node_color = ["blue" if node in top_pagerank_addresses else "gray" for node in subgraph.nodes()]

# Plot Graph
plt.figure(figsize=(12, 8))
nx.draw(subgraph, pos, node_size=node_size, node_color=node_color, edge_color="gray", alpha=0.6, with_labels=False)

plt.title("PageRank - Most Influential Wallets", fontsize=14)
plt.axis("off")
plt.show()
```

Visualize

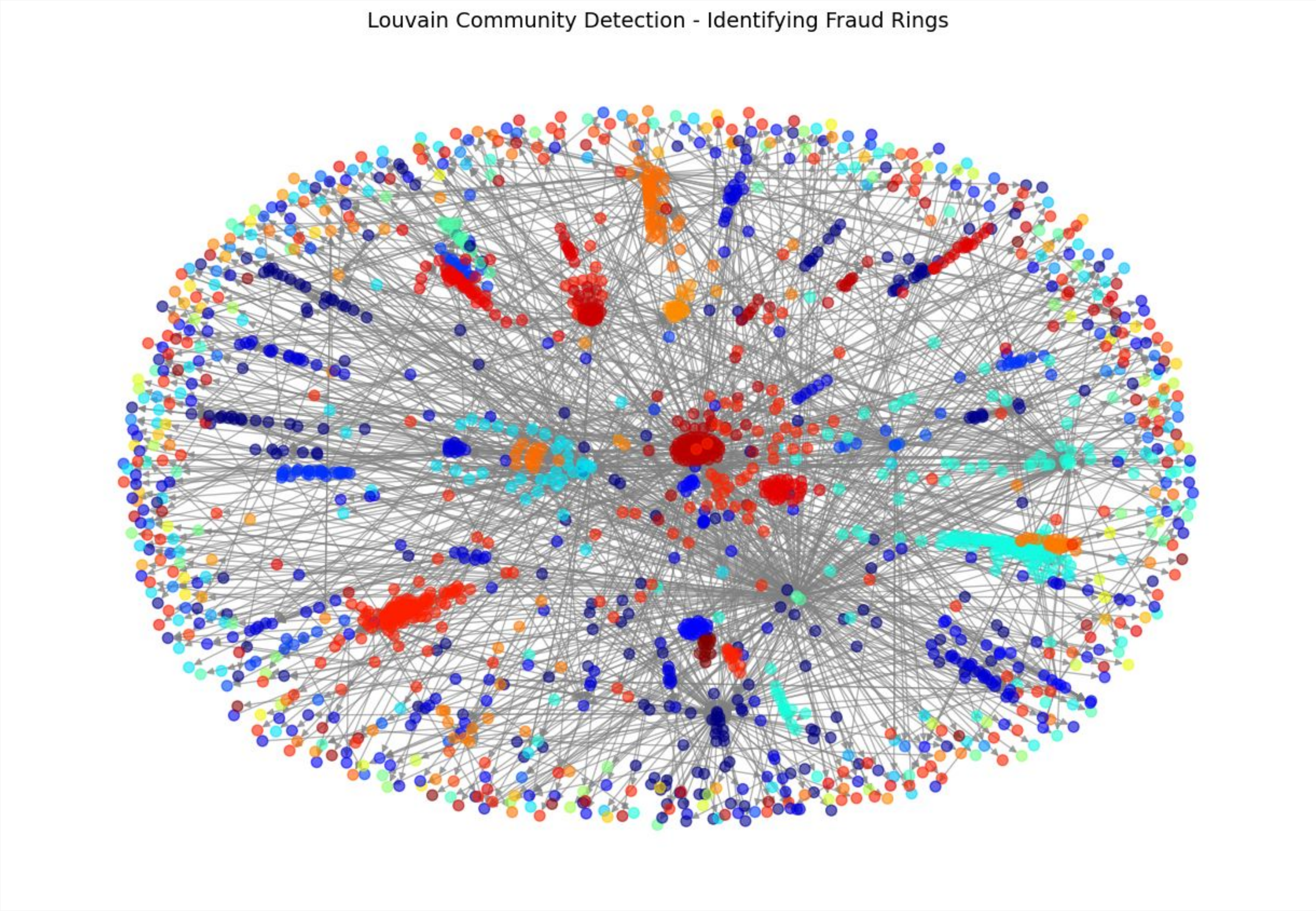


Project Progress - Clustering

Performed Louvain clustering to detect tightly connected wallet communities and visualize potential fraud rings.

```python
# Extract unique community IDs
communities = set(partition.values())

# Assign colors to communities
community_colors = {comm: plt.cm.jet(i / len(communities)) for i, comm in enumerate(communities)}

# Define node colors based on their community
node_color = [community_colors[partition[node]] for node in G.nodes()]

# Draw the network
plt.figure(figsize=(12, 8))
pos = nx.spring_layout(G, seed=42)  # Fix layout for consistency
nx.draw(G, pos, node_color=node_color, edge_color="gray", node_size=50, alpha=0.6, with_labels=False)

# Add title
plt.title("Louvain Community Detection - Identifying Fraud Rings", fontsize=14)
plt.axis("off")
plt.show()
```

Result



What's Next?

Objective

Use binary classification to detect high-risk (scam-related) wallet addresses in transaction networks.

Method

Implement Graph Neural Networks (GNN) using GCNConv to learn patterns from address connectivity.

Rationale

High-risk wallets often show structural traits—many connections, suspicious flows, and links to scam wallets.

Advantage

GCN captures network context, enabling better detection of subtle fraud patterns than traditional models.

Thanks

CREDITS: This presentation template was created by **Slidesgo**, and includes icons, infographics & images by **Freepik**