CSCI 5253: Project Report
Detecting Fraudulent Activity in Bitcoin Transactions

Suma Dodmani, Prathyusha Gayam, Poorwa Hirve, Swathi Upadhyaya

***Abstract*— The increasing use of Bitcoin as cryptocurrency goes hand in hand with the increase in cybercrime activities. In this project, we attempt to study the Blockchain network and identify fraudulent activities based on the Bitcoin transactions. We use a datacenter-scale approach on actual Bitcoin data in order to scale to a large dataset using Neo4j graph database to store the data as well as preserve the relations as in real blockchain along with PySpark. We run K-means clustering on the generated Neo4j graph to find out the fraud transactions.**

***Index Terms*— Bitcoin, Blockchain, Neo4j, PySpark, K-means clustering**

I. Introduction

Bitcoin protocol has been one of the significant protocols ever since the technology was made known to the world and bitcoin along with blockchain is considered strong cryptographically. However, over the years there have been several cases of attacks and fraud transactions. Here, fraud transactions refer to a transfer of funds to a destination not authorized by the legitimate owner [1]. There is a lot of literature stating methods to prevent such transactions from occurring but time and again there have been cases reported as discussed in [2]. In such cases, its important to know the features of these transactions as the ready availability of all transaction information in bitcoin network makes it easy to trace the flow of money from compromised accounts in the wake of reported heist or robbery [3]. In this project we aim to analyze the blockchain and bitcoin network to identify the fraudulent transactions by analyzing the transaction features.

Transaction is the central aspect of bitcoin blockchain. And for our problem, a clear understanding of the characteristics and traceability of transactions forms a major part. A transaction is transfer of a certain bitcoin value which is then broadcasted to the entire network and then collected into blocks. Transaction holds information about the origin of funds and address of destination. It is possible to view every transaction that has ever been collected into the block. This makes

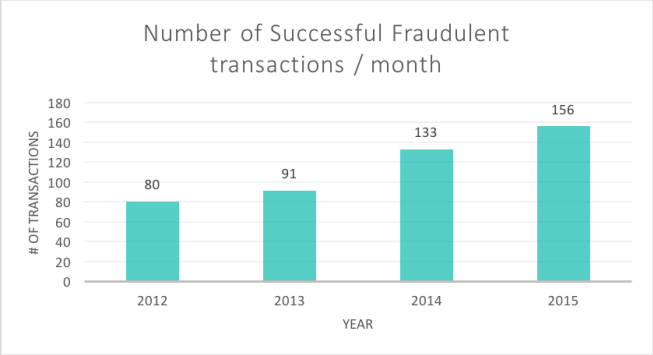


Fig. 1. Increase in successful Bitcoin fraud transactions from 2012 to 2015 [8]

it important to ensure the safeguarding the privacy of the wallet information of the users. The whole transaction file is digitally signed with a private key by the user sending the funds. Signature along with public key are enclosed in the transaction. This allows anyone to validate the transferred Bitcoins if they are owned by the sender [4]. Viewing linkage of multiple blocks involved is possible as the hash value of previous block is present in the new block created. This acts as a link and makes it part of the link.

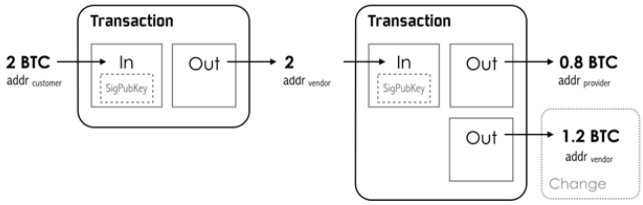


Fig. 2. An example of two transaction

Source: http://tech.eu/features/808/bitcoin-part-one/

Figure 2 shows a simple bitcoin transaction. It has two transactions; the first transaction gets 2 BTC from a customer along with public key of the customer. The 2BTC received in transaction 1 is sent to another transaction (Transaction 2) and the amount received here is sent out the producer (0.8 BTC) and the change

(1.2 BTC) from transaction is returned to the vendor. Traceability of all transactions made by public keys shifts the onus of maintaining anonymity to the users of the public keys [3].

Bitcoin robberies cannot be prevented in totality. And there have been many cases as listed in [2]. The appalling fact is each type of robbery varies in characteristic with others. Thus, it is important to identify the features of these transactions as behavior of fraudulent transaction is certain to be different from the normal transaction. Fraud patterns are more complex and requires a lot of pattern analysis and discovery [4] and this is best done in Neo4j as it preserves the relations between components as in real network.

Bitcoin Blockchain data is well-researched. Since it has been gaining attention, the data is preserved and made available from the time the technology came into existence. Thus, owing to the fact the data is humongous, processing it in any form will need great computing capabilities. We use Neo4j to store the bitcoin blockchain data and use spark to apply K-Means clustering on the data.

Bitcoin data is unlabeled thus the best approach is clustering based on relevant features for outlier detection. We apply k-means clustering to achieve this. K-means provides basis that outliers will essentially be found farthest from the centroids of the clusters they are associated with. Euclidean distance of each object from the calculated centroid provides us with the measure of how far the object is from the center. Thereby this provides the information of whether an object is classified as an outlier [5].

We begin by stating the previous work related to detecting the fraudulent transactions in Section 2. Further, the Section 3 provides an overview of the system architecture where we also provide details about the data and discuss the algorithm and features considered. Further in Section 5, we discuss the results obtained and the factors which played a major role in detecting fraud transactions. Our architecture and machine learning algorithm were successful in identifying fraudulent transactions based on the features we extracted. Figure 6 and Figure 7 show our results on the datasets we tested on.

II. Related Work

In this section we look into prior work on bitcoin blockchain and fraudulent transactions.

A. Unsupervised Learning for Robust Bitcoin Fraud Detection

This paper [1] aims to find and classify anomalies in the Bitcoin network. These anomalies are observed based on the transaction patterns. Observation of the patterns aids in detecting fraudulent activities. Once these anomalies are identified, the paper also seeks to assess the performance of the algorithms using publicly available Bitcoin transaction data.

The paper implements this on 37,450,461 transactions using R programming. The dataset they used was `http://compbio.cs.uic.eduldatalbitcoin/`

B. Analysis of Bitcoin Network Dataset for Fraud

This paper [3] also aims to detect fraud in the Bitcoin network as previous paper. They observed that using k-means clustering, clusters with significantly less number of nodes can be identified as containing fraudulent transactions.

The dataset contained a total number of 881,678 nodes and 1,617,212 edges. The dataset was obtained by crawling `www.blockchain.info` and publicly available datasets `http://anonymity-in-bitcoin.blogspot.com/2011/09/code-datasets-and-spsn11.html`

C. Proposed work

To visualize the Bitcoin transactions, we decided to use graph databases. Using the graph database, we can preserve the relations as in actual blockchain network. Once we set up the blockchain network, we apply k-means clustering as discussed in [1][2], however differ in the features considered and we do this specific to the transaction nodes as our aim to to detect fraudulent transactions. [3] talks about considering clusters with significantly less number of nodes to contain fraud transaction. However, we also observe that even if the fraud transactions belong to bigger clusters, they are very far from their centroid as opposed to the other members of the cluster. We use datacenter-scale computing technologies in order to scale to large data and parallelize our process. We used **ELTE** bitcoin dataset and the technological stack used includes **PySpark** through Amazon EMR, **Amazon S3** for storage of data, **Neo4j** graph database on Amazon EC2. This is further discussed in the system design.

III. System Design

A. Dataset

The model is trained on **ELTE** Bitcoin dataset that has the following files:

- **Blockhash.txt:** enumeration of all blocks in the blockchain, **(34MB)**
  - block_id - block ID
  - b_hash - block hash (identifier in the blockchain)
  - prev_block - previous block ID
  - b_time - block creation time
  - no_of_ts - number of transactions in the block
- **txhash.txt:** transaction ID and hash pairs **(8.6GB)**
  - tx_id - transaction ID
  - txhash - transaction hash
- **tx.txt:** enumeration of all transactions with 4 columns **(3.4GB)**
  - tx_id
  - block_id
  - n_ips - number of inputs
  - n_ops - number of outputs
- **txin.txt:** list of all transaction inputs (sums sent by the user - **8.2GB**)
  - tx_id - transaction ID
  - input_id - input ID
  - ip_addr_id - address ID, sending address
  - ip_value - value, integer sum in **Satoshis (1e-8 BTC)**
- **txout.txt:** list of all transaction outputs (sums received by the users - **9.4GB**)
  - tx_id
  - output_id - output ID
  - op_addr_id - address ID , receiving address
  - op_value - Value, integer sum in Satoshis (1e-8 BTC)

B. Data Preprocessing

Since our original dataset was huge **(~29GB)** we first reduced the dataset size to **~8GB**. Our processed dataset has **30** fraudulent transactions. For evaluation, we further divided this reduced dataset to **~20 MB**. This small dataset had **6** fraudulent transactions.

C. System Architecture

We use Amazon EC2 for scalability in cloud. In order to achieve parallelization, we read data from **.csv** files stored in Amazon S3 buckets and load into Resilient Distributed Datasets (RDDs). Storing the data into PySpark RDD ensures that RDD is divided into logical partitions which may be computed on different nodes of the cluster. This data is further fed to Neo4j to create the graph and perform clustering [6].

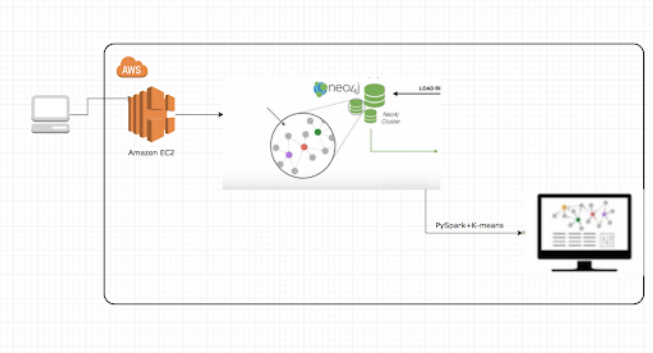


Fig. 3. System Architecture

D. Procedure for Setup and Execution

1) Create an EMR cluster enabling Spark with 1 master and 4 slaves. Choose a large instance (e.g. **r4.2xlarge**) as large dataset will be handled.
2) Create an EC2 instance and install Neo4j on it. Steps to deploy Neo4j using a **.tar** file have been uploaded to our Git repository. [9]
3) Upload dataset to S3 bucket.
4) Execute the Python code (again from our Git repository) on the data from the EMR cluster using **spark-submit**.
5) The updated graph database along with the centroids can be viewed from the Neo4j browser.

E. Decisions taken before finalizing the above procedure

1) The dataset was finalized to ELTE after viewing many different types of datasets and their contents.
2) We decided on r4.2xlarge instances after experimenting with many instances and getting memory issues.
3) Amazon AWS was decided for Neo4j because of its ease of use as compared to Google Cloud. S3 buckets were chosen to store large amounts of data rather than using **scp** and using up space on the instance.
4) We were able to run Neo4j queries on AWS instance, but using Spark would help in parallelizing it and make the process more efficient, not to mention handle data effectively along with S3.

5) We tried to use Spark and Neo4j with Scala as Spark is natively built on Scala. But this did not work hence we finalized PySpark.
6) Initially, we set up Spark on EC2 manually. However, we were running into issues with PySpark and reading from S3 buckets due to the version problems of Hadoop. Hence, we decided to use EMR, along with the ease of use and creation of master-slave nodes.
7) Neo4j through Amazon Marketplace did not allow us easy access to configuration files, so we downloaded a **.tar** file to set up Neo4j manually.
8) K-means clustering
   - Initially we considered only centroid based clustering method where outliers (fraud transactions) form a small cluster to. However, upon examining we found that there were a few fraud transactions farthest from the centroid in bigger cluster. Thus, we considered both distance based metric and cluster size to identify fraudulent transactions.
9) Through experimenting, we found out that K-means clustering with $k = 3$ and iterations = 5 gave us relatively better results. We were successful in identifying fraudulent transactions based on the features we used. We chose 3 optimum features by comparing the features of bitcoin blockchain from [1] and [3].The features we were able to derive from the dataset are:
   a) **In - degree of transaction node:** The number of input transactions
   b) **Out - degree of transaction node:** The number of output transactions
   c) **Transaction value**: The sum of transaction values on the edges connecting to output nodes

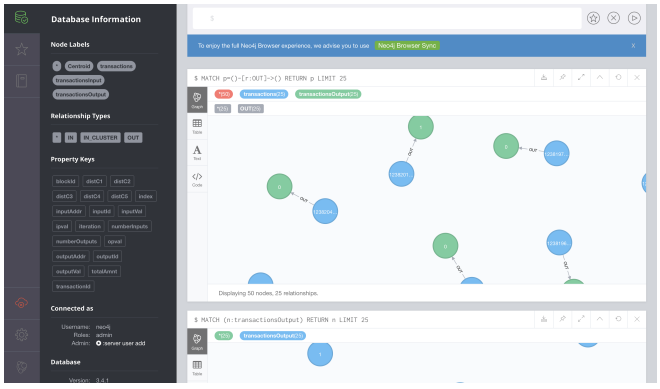


Fig. 4. Neo4j Browser

IV. Challenges and Difficulties

- Being novice to the field of Bitcoin and Blockchain Network there was need for lot of research and learning to familiarize ourselves with the concepts. Also, there was some degree of efforts towards understanding the Neo4j framework as a tool to construst Bitcoin Blockchain.
- Understanding the Data - We had two datasets to explore:
  1) Kaggle bitcoin blockchain - the live historical bitcoin blockchain data
     - BigQuery
     - Some of the data must be extracted from the hash values (like the amount sent etc.)
     - We realized that the data is not public. Moreover, the size of the data is 800 GB which would require us to have a lot of credits on Google Cloud and computational capabilities as well.
  2) The latest blockchain dataset on the ELTE project website.
     - All of our depiction of the model on Neo4j is done using this dataset.
     - We contacted the proprietors of the ELTE dataset and verify if the datatset contained any fraudulent transactions.
- Data Preprocessing
  - Modelling the large data into a graph and visualizing it required too much of processing power and memory. Also, the graph data model of Bitcoin Blockchain network requires twice the size of the actual dataset. So, we had to cut down the dataset to have 100,000 transactions.
- Configuring system architecture
  - The default storage option provided by AWS was not suffient for our model. This made us try different instances on AWS with varying storage size that would suit our requirements.

V. Evaluation and Findings

A. *Clustering: Centroid based K-Means Clustering*

- After building the complete graph, we applied K-Means clustering on the transaction nodes. We started the implementation by considering 3 centroids ($k = 3$) having random values. We ran three iterations through which we updated the centroid values such that the difference within the cluster

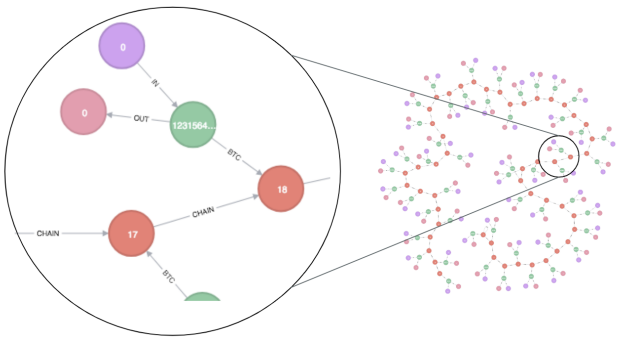
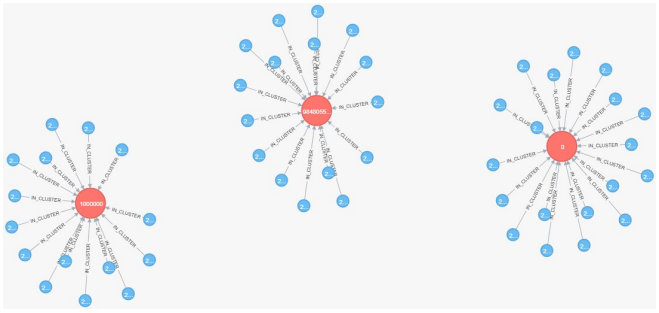


Fig. 5. Design of the Bitcoin-Blockchain Network in Neo4j

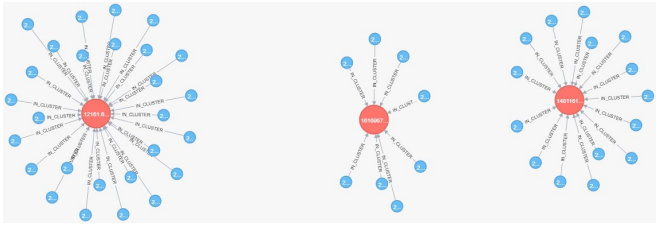
nodes is minimum and difference among the clusters is maximum. The clusters were created based on the following features of transaction nodes: in degree, out-degree and sum of transaction values on the edges connecting to output nodes.

- The size of the data considered for this is 20MB approximately. Table I shows the results.

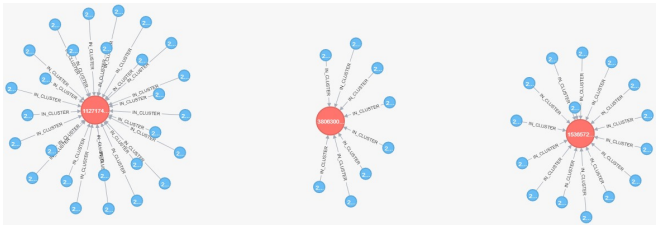
Cluster results after iteration 1:



Cluster results after iteration 2:



Cluster results after iteration 3:



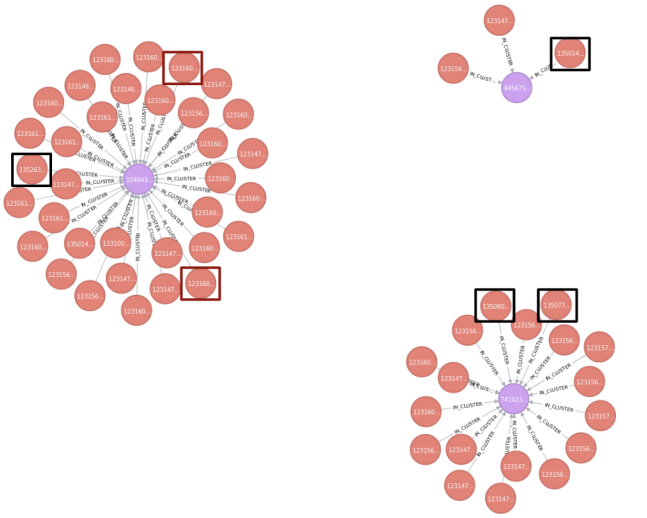


Fig. 6. Results on small Dataset

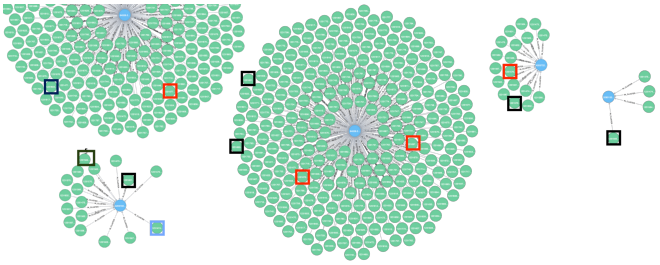


Fig. 7. Result on 100,000 transactions

Fraudulent Transactions:

- One of the important part is to understand the fraudulent transactions. In order to know more about this, we studied robbery cases, to try and find transaction details of involved parties.
- We wanted to confirm if the dataset we are investigating has captured these fraudulent activities, thus we also contacted one of the person who was part of the ELTE Bitcoin project, (Daniel Kondor) , who gave us a few more insights into the dataset and how possibly we could find fraudulent transactions.
- Thus, for the project, we depend exclusively on the reported robberies for the feature design. We assume that the [8](Bitcoin Robberies) is the true account of reported robberies.

The fraud transactions are highlighted by rectangles in Figures 6, 7.

- Red denotes that it is a fraud transaction but has not been detected by our algorithm.
- Black denotes that is is a fraud transaction and has

TABLE I

OBSERVED RESULTS OF K-MEANS CLUSTERING WITH $k = 3$

K-Means Results			
Iteration	Cluster-1	Cluster-2	Cluster-3
1	In degree: 1 Out Degree: 1 Transaction value: 0 Number of Nodes: 19	In degree: 1 Out Degree: 1 Transaction value: 10^6 Number of Nodes: 15	In degree: 1 Out Degree: 1 Transaction value: 9848055x10^3 Number of Nodes: 16
2	In degree: 0 Out Degree: 1 Transaction value: 12161.68 Number of nodes: 27	In degree: 0 Out Degree: 1 Transaction value: 14811619019.935 Number of nodes: 15	In degree: 0 Out Degree: 1 Transaction value: 1616967060.87 Number of nodes: 8
3	In degree: 0 Out Degree: 1 Transaction value: 11271740.185 Number of nodes: 27	In degree: 0 Out Degree: 1 Transaction value: 15365726954.6 Number of nodes: 15	In degree: 0 Out Degree: 1 Transaction value: 38063x10^5 Number of nodes: 8

been detected our algorithm.

Therefore, **4** out of **6** transactions have been correctly identified in the small dataset.

VI. REVIEW OF TEAM MEMBER WORK

A. Combined Efforts

Most of our efforts in this project were combined as the topic and technologies were new to us and it required us to learn together as a group.

1) **Bitcoin Blockchain:**
   We understood the Bitcoin and Blockchain network in order to define our problem statement effectively.
2) **Obtaining the dataset:**
   We searched Kaggle, read papers and scoured the web before we decided on the ELTE dataset.
3) **Neo4j:**
   - We started out with studying Cypher Query Language (CQL)
   - Researched and followed tutorials to use our dataset and mapping our data to Neo4j to build the Blockchain network.
   - Studied on how to implement K-means clustering in Cypher [7]
   - Experimenting with the k value and number of iterations to get the optimum result.

In addition to working together, we split into two teams and worked as described below.

B. Suma & Swathi

- Swathi and Suma intially installed Neo4j on their local machine to learn Cypher queries and build graphs.
- Suma & Swathi's AWS accounts were used completely for the duration of the project.
- Suma and Swathi figured out the set of Cypher queries to build the Bitcoin Blockchain Network in Neo4j from the datasets.
- Suma took the initiative to contact the one of the people involved in the ELTE Bitcoin project (Daniel Konder) in order to obtain insight into the fraudulent transactions for verifying the results.
- Swathi spent much time learning Scala to use with Spark. She was close to getting it to work completely but we were running out of time so we couldn't implement it.
- Suma & Swathi converted the Cypher queries to use in PySpark and investigated the properties of fraudulent transactions and studied the known robbery cases to checkif our dataset has any fraudulent data. They also preprocessed the dataset, converting it into the reduced and the further reduced dataset. As mentioned in V, these reduced datasets depend on the reported robberies for the feature design.
- Suma spent much time on Google Cloud to try and upload our dataset. But she decided that AWS and S3 buckets for storage were the way to go as we were more familiar with AWS and had enough credits to do so.

C. Prathyusha & Poorwa

- Prathyusha was also involved in building the cypher queries to build Bitcoin Blockchain Network in Neo4j.
- Prathyusha & Poorwa designed the final working of the architecture together by experimenting

and figuring out what worked best. They first used Neo4j from Amazon Marketplace, but the configurations were difficult to access. Next, they researched and found a way to install the Neo4j server on a bare-metal EC2 instance which was the remote Neo4j server. They were responsible for setting up EMR cluster and configuring it to establish communication with Neo4j server. They then worked on the Python code to connect to the database from the cluster and run the queries. They worked parallelly with Swathi and were able to get PySpark and Neo4j working seamlessly so we went along with PySpark instead of Scala.

- Prathyusha, Swathi & Suma figured out the Cypher queries for K-means clustering. She developed the initial code for iterations and for the calculating the centroids.
- Poorwa configured AWS for Neo4j and PySpark. She figured out how to configure the **bolt** protocol for communication between PySpark and the Neo4j database.
- Prathyusha & Poorwa experimented with the k values and number of iterations for K-means clustering, finally deciding on $k = 5$ and iterations = 5 for 100,000 transactions as in Figure 7.

VII. Conclusions

Kmeans Clustering groups all the transactions with similar features into one cluster but it is not exclusively used for anomaly detection. For the problem at hand, we expect the fraud transactions to be spaced away from the normal transactions in euclidean plane as they differ in features we considered. In addition to considering in-degree, out-degree, and transaction-value as features considering other information like rate at which transactions are made would help us identify the fraud transactions better.

References

[1] Deepak Zambre, Ajey Shah, "Analysis of Bitcoin Network Dataset for Fraud", 2013

[2] https://bitcointalk.org/index.php?topic=83794.0post_ubitex_scam

[3] http://tech.eu/features/808/bitcoin-part-one/

[4] https://neo4j.com/blog/fraud-prevention-neo4j-5-minute-overview/

[5] P. Monamo, V. Marivate and B. Twala, "Unsupervised learning for robust Bitcoin fraud detection," *2016 Information Security for South Africa (ISSA)*, Johannesburg, 2016, pp. 129-134.

[6] https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm

[7] http://heidi.morkland.org/2016/03/k-mean-clustering-in-neo4j/

[8] https://seon.io/resources/2017/05/02/tor-bitcoin-and-the-dark-net-marketplaces-ruined-the-fraud-landscape/

[9] https://github.com/CSCI5253-Fall2018/final-project-project-gayam-dodmani-upadhyaya-hirve-1.git