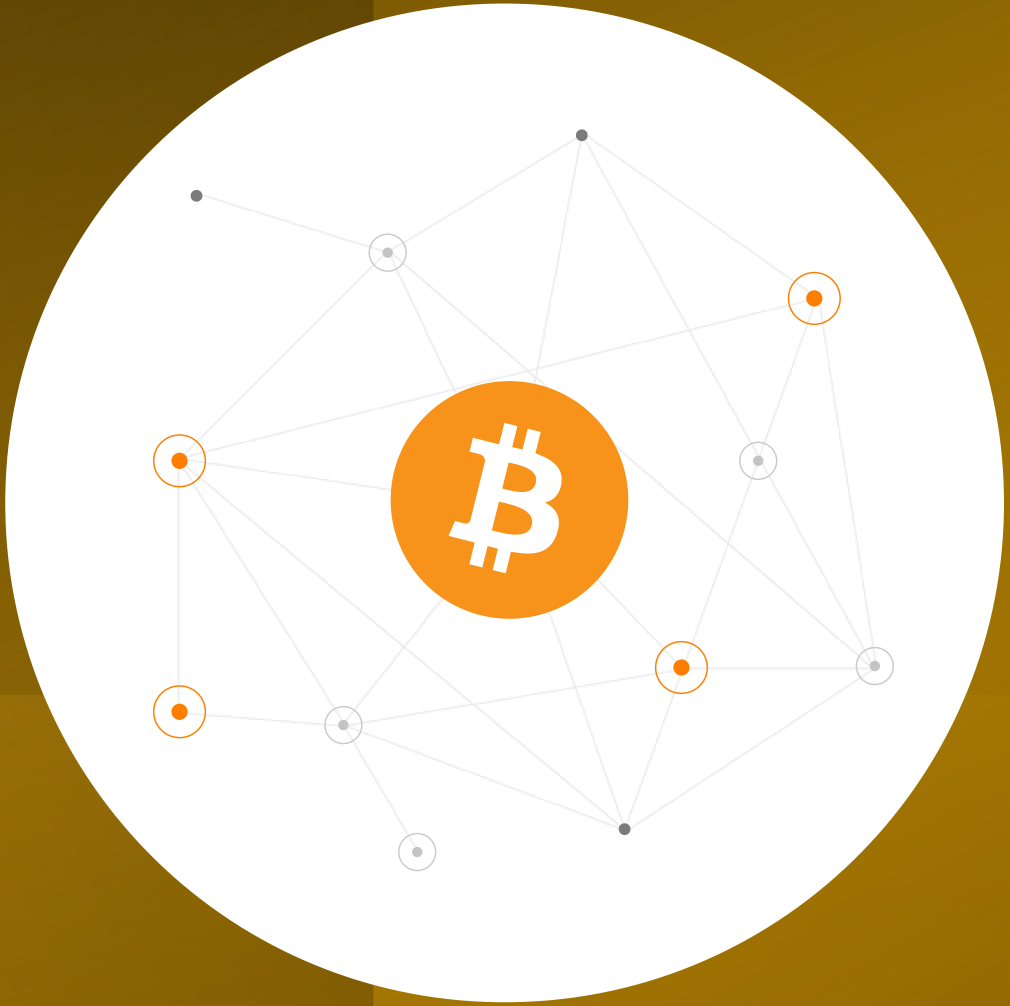
Bitcoin Address Clustering Algorithm Based on Multiple Heuristic Conditions

Vincenzo Imperati 1834930



Project Goal

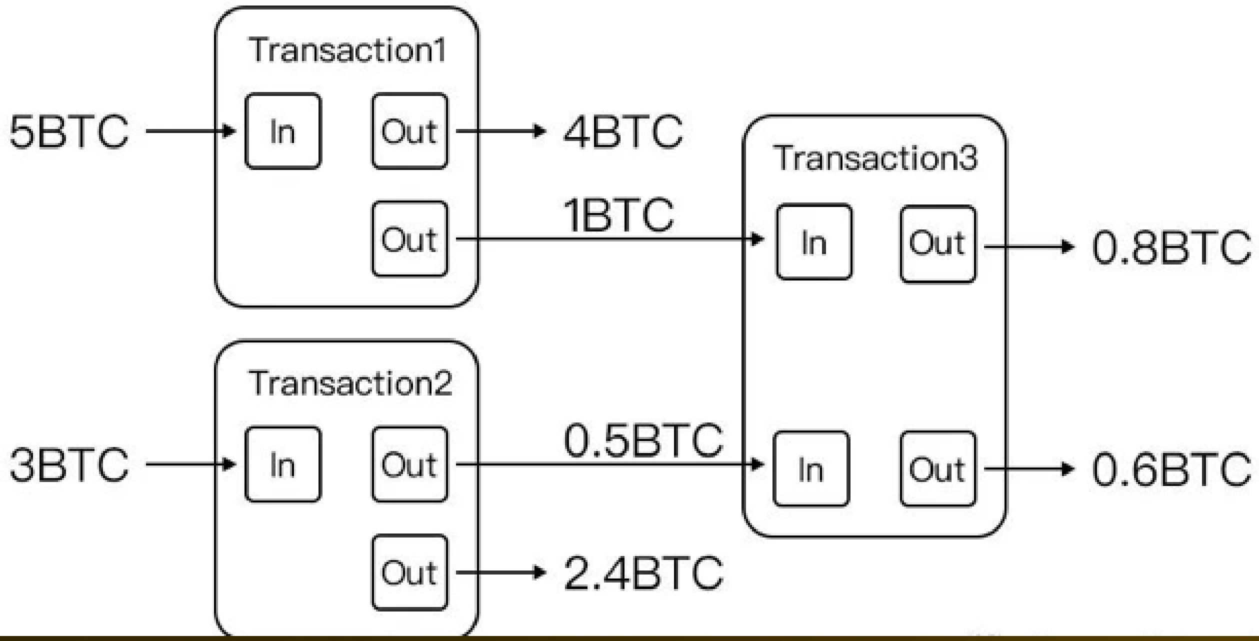
Extract informations from **Bitcoin blockchain**

Manipulate them to build **transaction graph**

Perform chain analysis tasks

Roadmap

Domain Overview: Concept of Transaction



A **transaction** is a transfer of Bitcoin value from Bitcoin addresses in input to Bitcoin addresses in output.

Transaction spent previews unspent transaction outputs (**UTXOs**) to create new unspent transaction outputs.

The balance of each bitcoin address is the sum of the value of its UTXOs.

Domain Overview: Trasparency

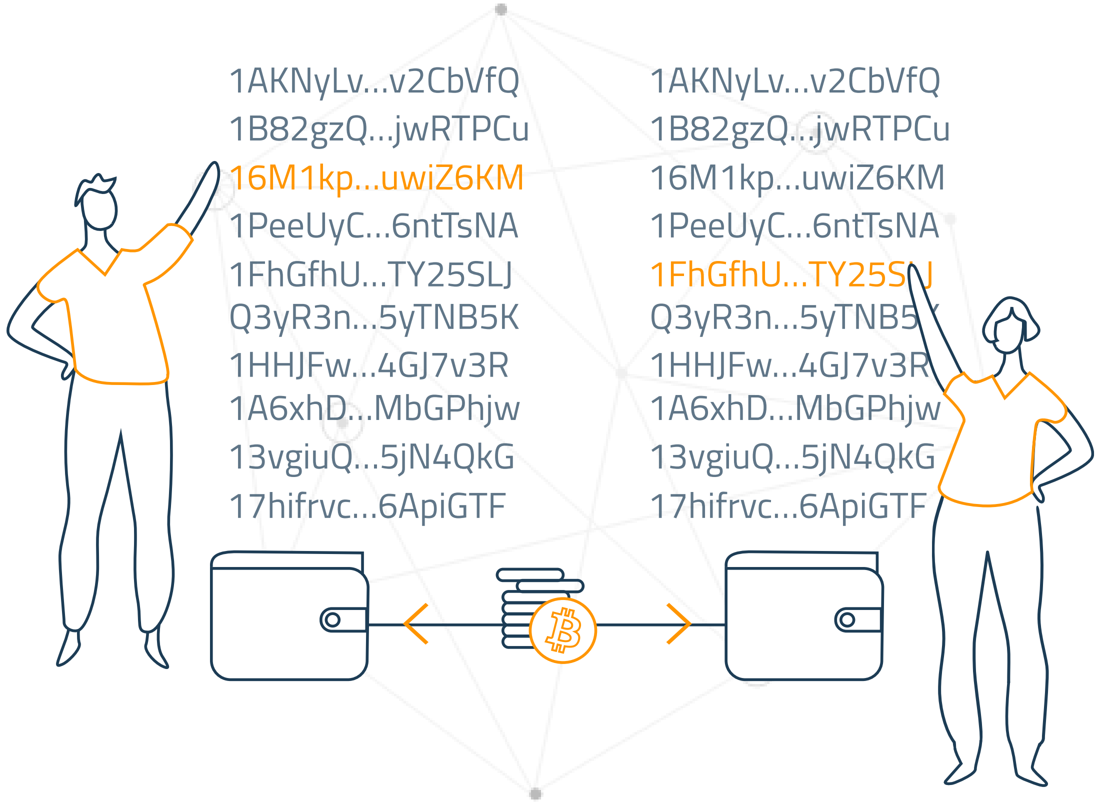
- All transactions are stored on the blockchain. These are **immutable** and **transparent**.
- It is possible to know **how many bitcoins** each address has and **how they are moved** from one address to another.

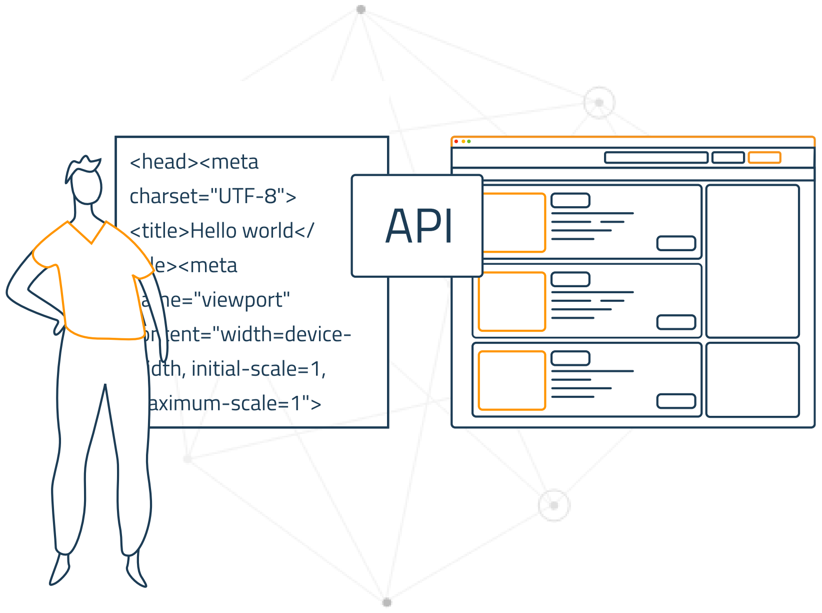
Domain Overview: Pseudo-anonimity

- In the the Bitcoin network, the only **unknown** information is to known **which person own which address**.

- If you know the address of a person, it is possible to know his movements.

- Through chain analysis, given an address of an entity, it is possible to obtain **all its other addresses**.





```
<head><meta
charset="UTF-8">
<title>Hello world</
le><meta
a he="viewport"
on ent="width=device-
idth, initial-scale=1,
aximum-scale=1">
```

Data Collection

- The first **115,000 blocks** of the Bitcoin blockchain, containing a total of **367,217 transactions**, were downloaded from the Blockchain.com site.

Data Processing: Blockchain Parsing

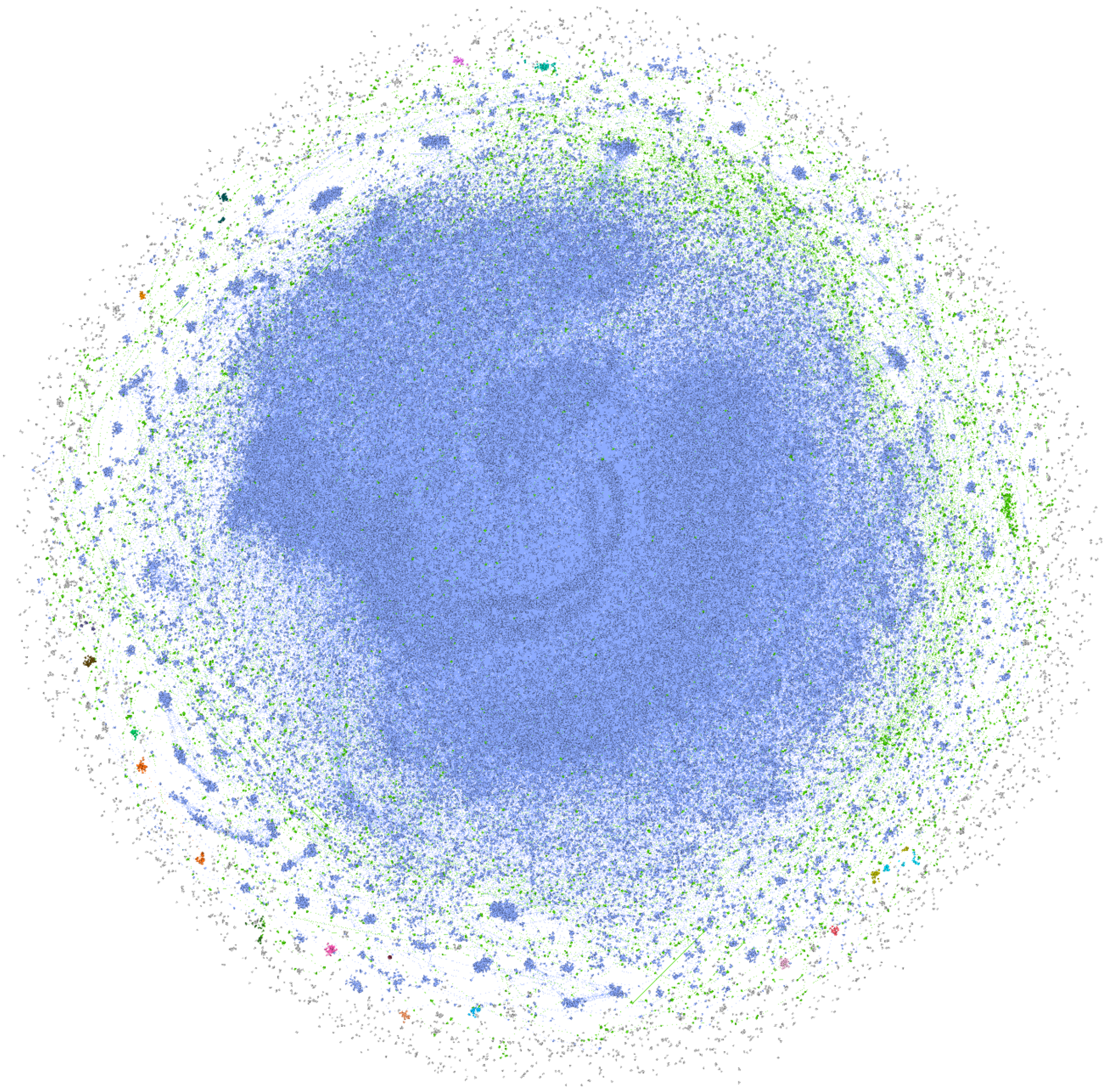
For each **raw block** are extrapolated usefull informations related to transactions and UTXOs:

- **Transaction**
  - id, tx_hash, block_height, block_hash, fee, n_input, amount_input, n_output, amount_output, temporal_index.
- **UTXO**
  - src_id, dst_id, src_position, dst_position, address, value.

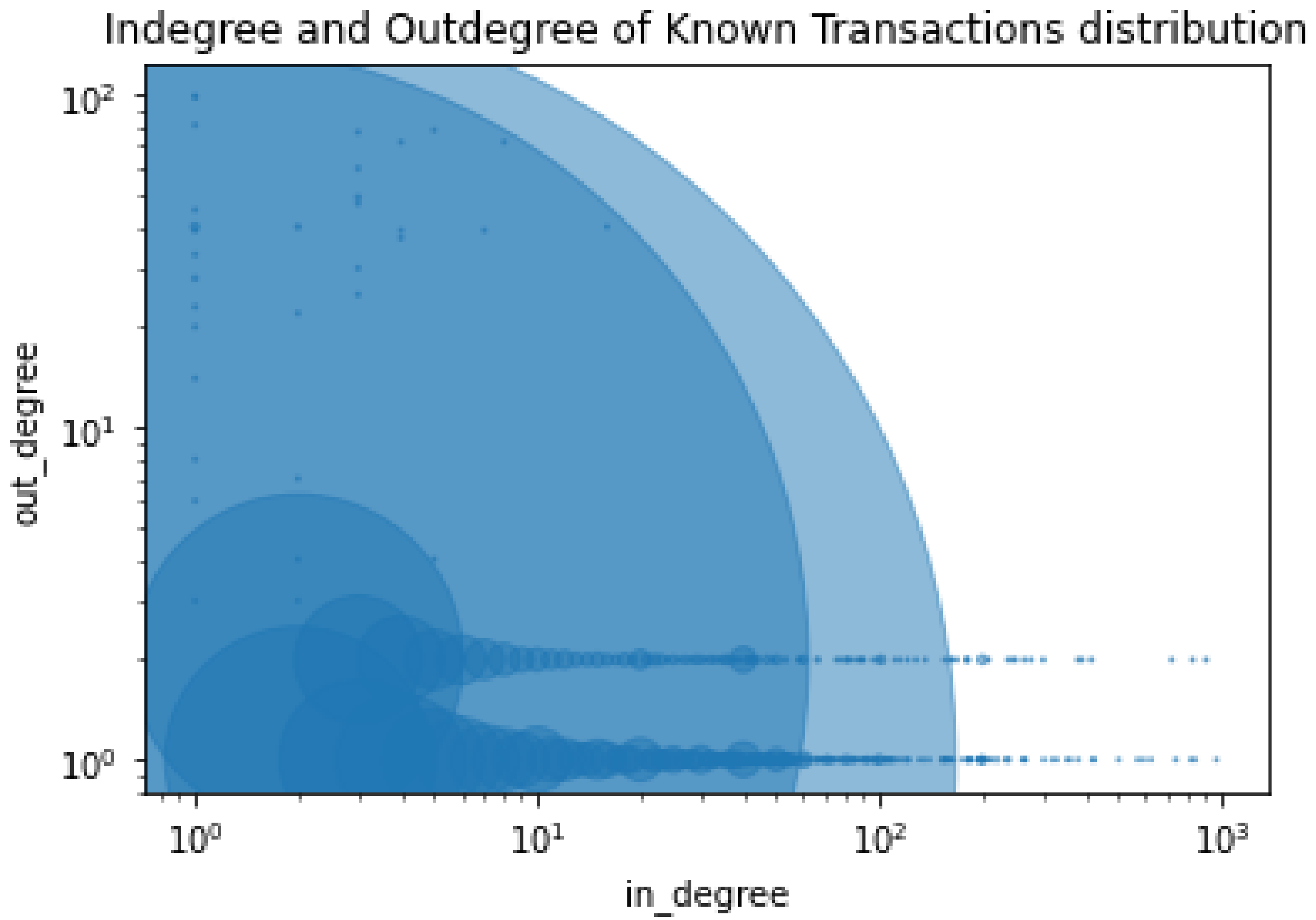
Data Processing: Transaction Graph

- Each transaction can be interpreted as a **graph node**.
- The transaction’s inputs are the **incoming edges**.
- The transaction’s outputs are the **outgoing edges**.
- The entire blockchain can be represented as a **multi directed graph**.
- Through these observations, the transaction graph was created starting from the downloaded blocks.
- The result is a graph of **559,528 nodes** and **630,301 edges**.

Data Processing: Transaction Graph



Graph Analysis: Indegree and Outdegree of Nodes



Heuristics: Multiple interpretations

There are at least nine possible interpretations:

1. Alice provides both inputs and pays 3 btc to Bob. Alice owns the 1 btc output (i.e. it is a change output).
2. Alice provides both inputs and pays 1 btc to Bob, with 3 btc paid back to Alice as the change.
3. Alice provides 1 btc input and Bob provides 3 btc input, Alice gets 1 btc output and Bob gets 3 btc output. This is a kind of CoinJoin transaction.
4. Alice pays 2 btc to Bob. Alice provides 3 btc input, gets the 1 btc output; Bob provides 1 btc input and gets 3 btc. This would be a PayJoin transaction type.
5. Alice pays 4 btc to Bob (but using two outputs for some reason).
6. Alice owns all inputs and outputs, and is simply moving coins between her own addresses (fake transaction).
7. Alice pays Bob 3 btc and Carol 1 btc. This is a batched payment with no change address.
8. Alice pays 3, Bob pays 1; Carol gets 3 btc and David gets 1 btc. This is some kind of CoinJoined batched payment with no change address.
9. Alice and Bob pay 4 btc to Carol (but using two outputs).

```
1 btc   ---->  1 btc
3 btc          3 btc
```

Heuristics: Satoshi

- It is estimated that Satoshi, in the first months of 2009, mined about 1,000,000 BTC.
- According to a New York Times research, we can assume the following: the owner of output addresses of a coinbase transaction is Satoshi with high probability, only if it is a coinbase transaction with **block_height < 19500**.
- (19500 block_height = July 14th 2009).

Heuristics:
Coinbase transaction mining address

- We can assume that the output address of a Coinbase transaction is controlled by the same entity (the miner that mined the block).
- There is **no reason** to send the block reward to non-proper addresses.

Heuristics: Common-input-ownership

- If a transaction has more than one input then **all those inputs** are owned by the **same entity**.

A (1 btc)	-->	X (4 btc)
B (2 btc)		Y (2 btc)
C (3 btc)		

- This transaction would be an indication that addresses B and C are owned by the same person who owns address A.
- One of the purposes of **CoinJoin** is to **break** this heuristic.

Heuristics: Single input and single output

- Transactions with a single input and a single output are interpreted as a **movement of funds** from one wallet to another under the control of the same owner.

A (1 btc) --> B (1 btc)

- This heuristic has a greater value when the funds moved from A to B equals the **entire balance** of wallet A.
- This heuristic fails if the transaction is actually a **payment without change**.

Heuristics: Consolidation transaction

- A transaction with multiple inputs and a single output is interpreted as a **consolidation transaction**, where all input UTXOs generate a single output UTXO.

A (1 btc) --> X (6 btc)
B (2 btc)
C (3 btc)

- This operation reveals that all transaction addresses belong to the same entity.
- This heuristic fails if the transaction is actually a **payment without change**.

Heuristics: Payment transaction with change address

- The **most common** transactions are those with one or more inputs and two outputs, where the two outputs represent the payment that A makes to B and the change of the payment that returns to A.

```
A (10 btc)      -->      X (6 btc)
                         Y (4 btc)
```

- We need to **identify the change** of the transaction.

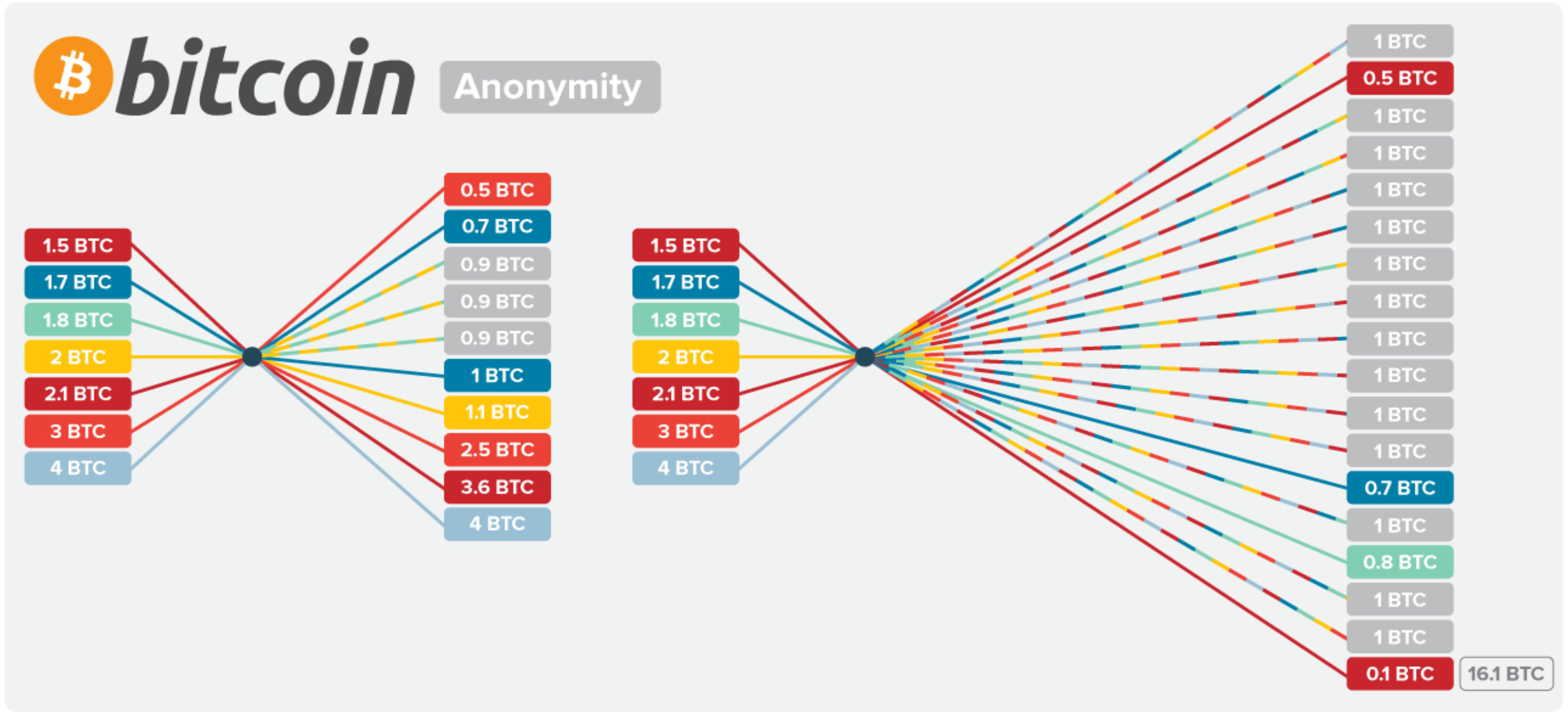
Heuristics: Change address detection

This heuristic has the objective to identify the change address in the payment transactions. To do this, different heuristics are used:

- same address in input and output heuristic
- address reuse heuristic
- unnecessary input heuristic
- new address in output heuristic
- round number heuristic

Heuristics: Mixed transaction recognition

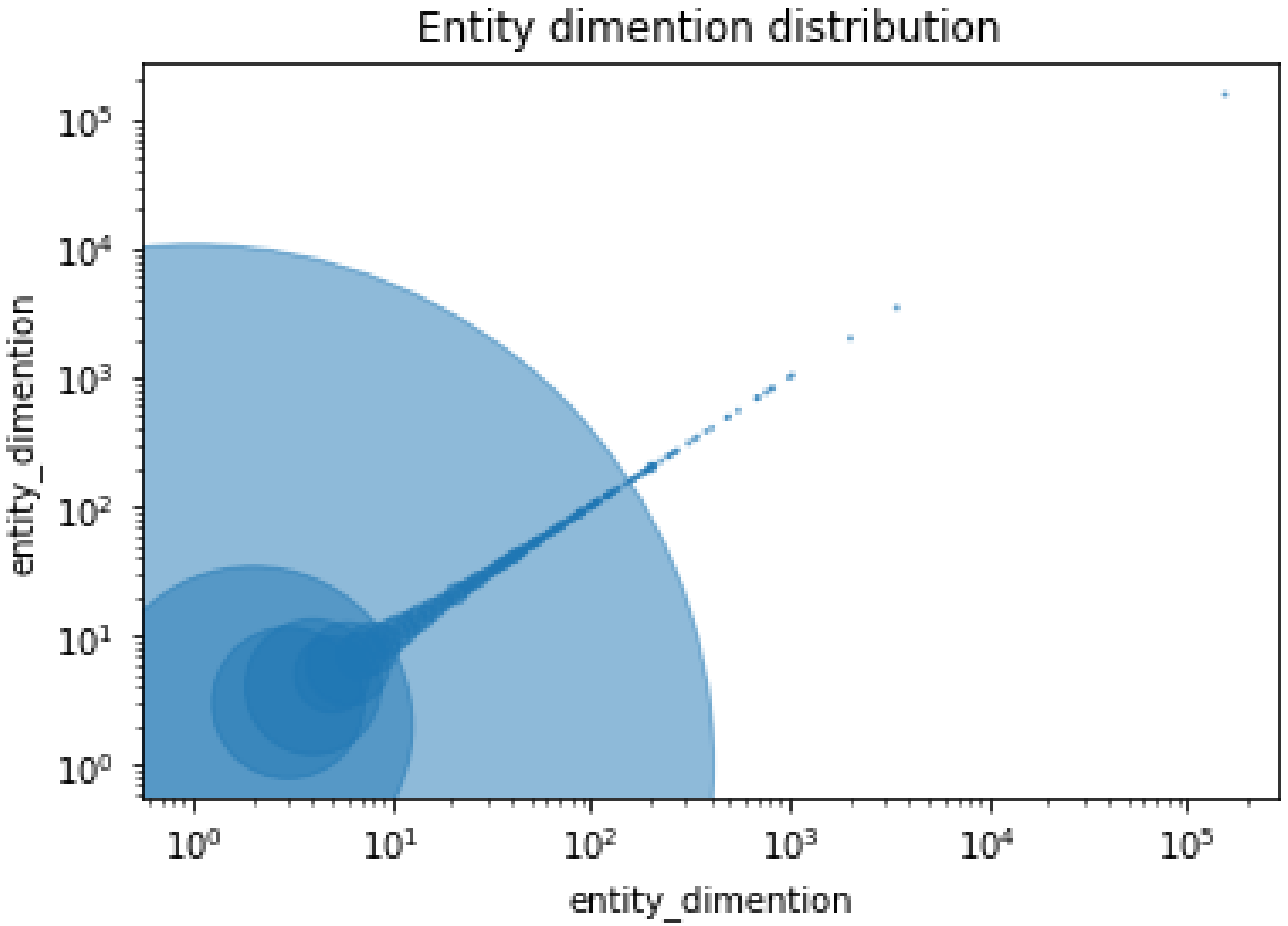
- Mixed transactions have many inputs and many outputs.
- Typically the outputs are characterized by the **same amount** of btc.
- These transactions **break the chain analysis**.
- On mixed transactions it is possible to apply **taint analysis** and **coinjoin sudoku**.



Clustering Algorithm Results

- In the first 115,000 blocks there are 334,177 different addresses which can therefore be assimilated to **334,177 different entities**.
- The algorithm clusters the addresses thanks to the heuristics described above, **reducing** the entities **to 109,074**.
- The algorithm reduced the entities by **67%**.

Clustering Algorithm Results: Entity Dimention



Clustering Algorithm Results: Observation

- The implemented clustering algorithm aims to **reduce false positives** as much as possible.
- This results in **many clusters populated by only one or two addresses**.
- Since it is unusual for a single entity to possess only one or two addresses, it is reasonable to assume that these small clusters can somehow be joined to larger entities through stronger clustering and more trivial analyzes.
- For this reason, in some use cases, if necessary it is possible not to consider small clusters, assuming that these are incorporated into larger clusters.
- Note that in chain analysis is **more important reduce false positives then false negatives**.

Use Cases: Visualize entity movements from an address

- Thanks to the results of clustering algorithm, a **web app** has been developed.
- Via the transaction graph, the web app displays all the movements made by an entity that owns the entered address.

Future Improvement

- Load the transaction graph in **Neo4j** for the best scalability.
- Find the **change addresses** iteratively **until convergence**.
- Implement **Coinjoin Sudoku**.
- Implement **new complex heuristics** after the observation of clustering algorithm results.
- **Merge** or **demerge clusters** thanks to **off-chain** informations.
- Apply **louvain community detection** algorithm after clustering algorithm.
- **Caching web app data** for increase scalability.