TxProbe: Discovering Bitcoin’s Network Topology Using Orphan Transactions

Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller and Bobby Bhattacharjee
WHAT DO WE KNOW ABOUT THE TOPOLOGY?
WHAT DO WE KNOW ABOUT THE TOPOLOGY?

Number of nodes and location of them
WHAT DO WE KNOW ABOUT THE TOPOLOGY?

Number of nodes and location of them

GLOBAL BITCOIN NODES DISTRIBUTION
Reachable nodes as of Thu Feb 07 2019
10:26:44 GMT+0000 (Greenwich Mean Time).

10365 NODES
24-hour charts »

Top 10 countries with their respective number of reachable nodes are as follow.

<table>
<thead>
<tr>
<th>RANK</th>
<th>COUNTRY</th>
<th>NODES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>United States</td>
<td>2570</td>
</tr>
<tr>
<td>2</td>
<td>Germany</td>
<td>1968</td>
</tr>
<tr>
<td>3</td>
<td>France</td>
<td>689</td>
</tr>
<tr>
<td>4</td>
<td>Netherlands</td>
<td>514</td>
</tr>
<tr>
<td>5</td>
<td>China</td>
<td>411</td>
</tr>
<tr>
<td>6</td>
<td>Canada</td>
<td>384</td>
</tr>
<tr>
<td>7</td>
<td>United Kingdom</td>
<td>355</td>
</tr>
<tr>
<td>8</td>
<td>Singapore</td>
<td>321</td>
</tr>
<tr>
<td>9</td>
<td>Russian Federation</td>
<td>277</td>
</tr>
<tr>
<td>10</td>
<td>Japan</td>
<td>228</td>
</tr>
</tbody>
</table>

More (100) »
WHAT DO WE KNOW ABOUT THE TOPOLOGY?

Number of nodes and location of them

The edges are hidden by design
WHY HAVE A HIDDEN TOPOLOGY?

An open topology could ease different types of attacks:

• Transaction deanonymization
• Network based attacks (e.g: Eclipse attacks)

The current approach of the Bitcoin Core is to keep it hidden
WHY HAVE AN OPEN TOPOLOGY?

We know nothings about how the network really is:

- Is the network decentralised?
- Are there supernodes controlling the network traffic?
  - Information withholding
  - Censorship
- Are there weak spots in the network that can be easily isolated?

Security by obscurity does not seem the proper way to go
How Bitcoin (Core client) nodes choose their peers?

- Pseudorandomly from the *addrman*
- **8 outbound** connections by default
  
  No pair of nodes in the same /16 (IPv4)

- **117 inbound** connection by default (no IP restriction here)

Bitcoin forks based on the Core client follow the same approach
BACKGROUND

Our inferring technique is based on transaction propagation.

We take advantage of how transactions are handled by nodes:

- orphans transactions
- double-spending transactions
Valid transactions are stored in mempool.

Transaction in mempool are eventually propagated throughout the node neighborhood.
Valid transaction are stored in **mempool**

**Transaction in mempool** are eventually propagated throughout the node neighborhood.
Valid transactions are stored in **mempool**.

**Transaction in mempool** are eventually propagated throughout the node neighborhood.

TRANSACTION PROPAGATION IN BITCOIN
Valid transactions are stored in mempool.

Transaction in mempool are eventually propagated throughout the node neighborhood.

TRANSACTION PROPAGATION IN BITCOIN

A

B

\[ \text{inv}(h(t_x_n)) \]

\[ \text{announce} \]

\[ \text{get}_\text{data}(h(t_x_n)) \]

\[ \text{request} \]

Wait for tx up to 2 min
Valid transactions are stored in mempool

Transaction in mempool are eventually propagated throughout the node neighborhood

A

B

\[ \text{inv}(h(t_x_n)) \]

\[ \text{announce} \]

\[ \text{get\_data}(h(t_x_n)) \]

\[ \text{request} \]

\[ tx(t_x_n) \]

\[ \text{deliver} \]

Wait for tx up to 2 min
A transaction is orphan if **some of the referenced UTXOs are unknown**

They can not be validated, so they are stored in a separated data structure known as **MapOrphanTransactions** (or **OrphanPool** for short)

Transactions in MapOrphanTransactions are **NOT forwarded to any node**

If the same transactions is offered again to the node (**inv message**), it will not requested back (**getaddr**)
DOUBLE-SPENDING TRANSACTIONS

id = 4F3…ED

Source: 4F3…ED
To: Bob

txB

B's mempool

(tx0

txn-1)

Source: 4F3…ED
To: Alice

txB'
DOUBLE-SPENDING TRANSACTIONS

A

id = 4F3…ED

B

Source: 4F3…ED To: Bob

Source: 4F3…ED To: Alice

B's mempool

txn-1

txn-0

txn-B

txB

txB'
DOUBLE-SPENDING TRANSACTIONS

A

Source: 4F3...ED

To: Bob

B

Id = 4F3...ED

Source: 4F3...ED

To: Alice

B's mempool

<table>
<thead>
<tr>
<th>tx0</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>txn-1</td>
</tr>
<tr>
<td>txB</td>
</tr>
</tbody>
</table>
DOUBLE-SPENDING TRANSACTIONS

A

id = 4F3…ED

Source: 4F3…ED

To: Bob

xB

Source: 4F3…ED

To: Alice

xB'

B's mempool

B

tx0

txn-1

xB
A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

Three transactions

Observation tool
(like coinscope)
A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

A

B

Three transactions

Observation tool
(like coinscope)
A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes
A  B

Observation tool
(like coinscope)

Three transactions

Parent tx
xF

Flood tx

id = 4F3…ED

Marker tx
xM

taxP

Parent tx
A BASIC TOPOLOGY INFERRING TECHNIQUE

Two nodes

A

B

Observation tool (like coinscope)

US

Three transactions

Parent tx

txP

Marker tx

txM

Flood tx

txF

id = 4F3...ED
POSITIVE INFERRING TECHNIQUE
POSITIVE INFERRING TECHNIQUE

A

US

B
POSITIVE INFERRING TECHNIQUE
POSITIVE INFERRING TECHNIQUE

A's Mempool
∅

B's Mempool
∅

B’s MapOrphanTransactions
∅
POSITIVE INFERRING TECHNIQUE

A's Mempool
∅

A

B

US

B's MapOrphanTransactions
∅

B's Mempool
∅

\(tx_P\)
(1)

\(tx_F\)
(1)
POSITIVE INFERRING TECHNIQUE

A's Mempool
∅

B's Mempool
∅

B's MapOrphanTransactions
∅
POSITIVE INFERRING TECHNIQUE

A's Mempool ⊊ txP (1)

B's Mempool ⊊ txF (1)

B's MapOrphanTransactions ⊊ ∅
POSITIVE INFERRING TECHNIQUE

A's Mempool

txP

(1)

B's Mempool

txF

(1)

B's MapOrphanTransactions

∅
POSITIVE INFERRING TECHNIQUE

A's Mempool

B's Mempool

A

B

US

A's MapOrphanTransactions

B's MapOrphanTransactions

txP (1)

txF (1)

∅
POSITIVE INFERRING TECHNIQUE

A’s Mempool

B’s Mempool

B’s MapOrphanTransactions

A

US

B

A’s Mempool

B’s Mempool

B’s MapOrphanTransactions

txP

txF

(1)

∅
POSITIVE INFERRING TECHNIQUE

A's Mempool

A

B's Mempool

B

US

A's MapOrphanTransactions

B's MapOrphanTransactions

txP

(1)

txM

(2)

txF

(1)

∅
POSITIVE INFERRING TECHNIQUE

A's Mempool

B's Mempool

US

A

B

B's MapOrphanTransactions

∅

B's Mempool

txF (1)

A's Mempool

txP (1)
POSITIVE INFERRING TECHNIQUE

A's Mempool
- txP (1)
- txM (2)

B's Mempool
- txF (1)

B's MapOrphanTransactions
- Ø
POSITIVE INFERRING TECHNIQUE

A's Mempool
- txP (1)
- txM (2)

B's Mempool
- txF (1)

B's MapOrphanTransactions
- ∅
**POSITIVE INFERRING TECHNIQUE**

A’s Mempool

- txP (1)
- txM (2)

B’s Mempool

- txF (1)

B’s MapOrphanTransactions

∅
**POSITIVE INFERRING TECHNIQUE**

![Diagram showing A's and B's Mempools and MapOrphanTransactions]

- **A's Mempool**
  - txP (1)
  - txM (2)

- **B's Mempool**
  - txF (1)
  - txM (3)

**US**
NEGATIVE INFERRING TECHNIQUE

A’s Mempool
∅

B’s Mempool
∅

B’s MapOrphanTransactions
∅
NEGATIVE INFERRING TECHNIQUE

A’s Mempool
Ø

B’s Mempool
Ø

B’s MapOrphanTransactions
Ø

A

B

US
txP (1)
txF (1)
NEGATIVE INFERRING TECHNIQUE

A's Mempool
∅

US

B’s Mempool
∅

B’s MapOrphanTransactions
∅
NEGATIVE INFERRING TECHNIQUE
NEGATIVE INFERRING TECHNIQUE

A's Mempool:
- txP (1)
- txM (2)

B's Mempool:
- txF (1)

B's MapOrphanTransactions:
- ∅
NEGATIVE INFERRING TECHNIQUE

A’s Mempool

B’s Mempool

B’s MapOrphanTransactions

\( \emptyset \)
NEGATIVE INFERRING TECHNIQUE

A's Mempool

B's Mempool

B's MapOrphanTransactions

\( \emptyset \)
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

A's Mempool
- txP (1)
- txM (2)

B's Mempool
- txF (1)

B's MapOrphanTransactions
- txM (3)

Negative edge inferring

A's Mempool
- txP (1)

B's Mempool
- txF (1)

B's MapOrphanTransactions
- ∅
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

\[ \text{inv}(h(\text{txM})) \]

A's Mempool

<table>
<thead>
<tr>
<th>txP</th>
<th>(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>txM</td>
<td>(2)</td>
</tr>
</tbody>
</table>

B's Mempool

<table>
<thead>
<tr>
<th>txF</th>
<th>(1)</th>
</tr>
</thead>
</table>

B's MapOrphanTransactions

\( \emptyset \)
Positive edge inferring

\[ \text{inv}(h(txM)) \]

A BASIC TOPOLOGY INFERRING TECHNIQUE
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

A

B

\( \text{inv}(h(txM)) \)

\( \text{get\_data}(h(txM)) \)

A's Mempool

- txP (1)
- txM (2)

B's Mempool

- txF (1)

B's MapOrphanTransactions
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

$(\text{inv}(h(\text{txM})))$

get_data$(h(\text{txM}))$

edge does not exist

A's Mempool

| txP (1) |
| txM (2) |

B's Mempool

| txF (1) |

B's MapOrphanTransactions $\emptyset$
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

inv(h(txM))

A's Mempool

| txP | (1) |
| txM | (2) |

B's Mempool

| txF | (1) |

B's MapOrphanTransactions Ø
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

$\text{inv}(h(txM))$

ε

A's Mempool

<table>
<thead>
<tr>
<th>txP</th>
<th>(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>txM</td>
<td>(2)</td>
</tr>
</tbody>
</table>

B's Mempool

| txF  | (1) |

B's MapOrphanTransactions

∅
A BASIC TOPOLOGY INFERRING TECHNIQUE

Positive edge inferring

\[ \text{inv}(h(\text{txM})) \]

\( \varnothing \)

\( \checkmark \) edge does exist

A's Mempool
- txP (1)
- txM (2)

B's Mempool
- txF (1)

B's MapOrphanTransactions
\( \varnothing \)
ITS NOT THAT EASY

A

B

US
Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture.
Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture.
Long story short, the technique will fail if we add an additional node to the picture.
Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture.

A's Mempool

A's MapOrphanTransactions

C's Mempool

C's MapOrphanTransactions

B's Mempool

B's MapOrphanTransactions

US

<table>
<thead>
<tr>
<th>A's Mempool</th>
<th>C's Mempool</th>
<th>B's Mempool</th>
</tr>
</thead>
<tbody>
<tr>
<td>txP (1)</td>
<td>txP (2)</td>
<td>txF (1)</td>
</tr>
</tbody>
</table>

B's MapOrphanTransactions

∅
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture.
Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture

A's Mempool
- txP (1)
- txM (3)

C's Mempool
- txP (2)

B's Mempool
- txF (1)

B's MapOrphanTransactions
∅
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture

A's Mempool
- txP (1)
- txM (3)

B's Mempool
- txF (1)

C's Mempool
- txP (2)

B's MapOrphanTransactions
- ∅
Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture.
ITS NOT THAT EASY

Long story short, the technique will fail if we add an additional node to the picture
Long story short, the technique will fail if we add an additional node to the picture.
MAKE THIS WORK IN A REAL NETWORK

### Isolation

**A**

txP

### Synchrony

**A**

txP

**B**

txF

### Efficiency

\( \approx O(n) \)

\( \approx O(\sqrt{n}) \)

\( n = \#\text{nodes} \)
MAKE THIS WORK IN A REAL NETWORK

Isolation

Synchrony

Efficiency

\[ \approx O(n) \]

\[ \approx O(\sqrt{n}) \]

\[ n = \#\text{nodes} \]
ISOLATION
ISOLATION

A

C

B

US

(2) txP
ISOLATION

A

B

C

US

A

C

(2) txP
ISOLATION
ISOLATION

\[ \text{GETDATA} \rightarrow \text{INV} \]

\[ \text{US} \rightarrow A \rightarrow B \rightarrow C \]

\[ (2) \]
ISOLATION
ISOLATION

A

B

C

US

GETDATA

TX

A

C
ISOLATION

HOW?
ISOLATION

HOW? INVBLOCKING
INVBLOCKING
INVBLOCKING

A

C

B

US
INVBLOCKING

![Diagram with nodes A, B, C, and US connected by INV(txP) relationships.]

- Node A
- Node B
- Node C
- Node US
INVBLOCKING

A

C

B

US
INVBLOCKING

Diagram showing relationships between nodes A, B, C, and US with transactions represented by "txP".
INVBLOCKING

A

B

C

txP

US
We have a 2-min window where isolation and synchrony are not a problem!
SIMPLIFIED TXPROBE

A's Mempool: ∅

B's Mempool: ∅

C's Mempool: ∅

C's Orphanpool: ∅

B's Orphanpool: ∅
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

B's Orphanpool

US
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

B's Orphanpool

A

B

C

US
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

A

B

C

US
SIMPLIFIED TXPROBE

- A's Mempool
- B's Mempool
- C's Mempool
- C's Orphanpool
- B's Orphanpool

Nodes:
- A
- B
- C
- US

Connections:
- A to C
- B to C
- US to C
- US to A
- US to B

No transactions found in mempools and orphans.
SIMPLIFIED TXPROBE

A's Mempool

B’s Mempool

C's Mempool

C's Orphanpool

B's Orphanpool

US
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

A

B

US
SIMPLIFIED TXPROBE

- A's Mempool: txP
- A's Orphanpool: txP
- B's Mempool: txP
- B's Orphanpool: txF
- C's Mempool: txF
- C's Orphanpool: ∅
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

A

B

C

US
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

A

B

C

US

txM

taxP

taxP

taxP

∅
SIMPLIFIED TXPROBE

A's Mempool

B's Mempool

C's Mempool

C's Orphanpool

B's Orphanpool

US
SIMPLIFIED TXPROBE

- A's Mempool: txP, txM
- B's Mempool: txP, txF
- C's Mempool: txP, txF
- C's Orphanpool: ∅
SIMPLIFIED TXPROBE

A's Mempool
- txP
- txM

B's Mempool
- txP
- txF

C's Mempool
- txF

C's Orphanpool
- Ø

B's Orphanpool
- Ø

US
SIMPLIFIED TXPROBE

A's Mempool
- txP
- txM

B's Mempool
- txP

C's Orphanpool
- txM

C's Mempool
- txF

B's Orphanpool
∅
TXPROBE - PROTOCOL OVERVIEW

- Choose a target node
- Create Parent, Marker and Flood transactions
- INVBLOCK the network
- Send Flood to all connected nodes but target
- Let Flood propagate
- Send Parent to target
- Send Marker to target
- Let Marker propagate
- Request marker back from all nodes but target
TXPROBE - PROTOCOL OVERVIEW

- Choose a target node
- Create Parent, Marker and Flood transactions
- INVBLOCK the network
- Send Flood to all connected nodes but target
- Let Flood propagate
- Send Parent to target
- Send Marker to target
- Let Marker propagate
- Request marker back from all nodes but target

For every node in the network
For a network like **Bitcoin mainnet**:

- **nodes**: \( \approx 10000 \)
- **time**: \( \approx 8.25 \) hours
- **cost**: \( 573210-764280 \) satoshi (5 sat/byte) \( \approx $(20-30) \)
We run 5 Bitcoin Core nodes as ground truth

We define our precision / recall by checking how well can we infer the ground truth nodes connections

Over 40 trials and with 95% confidence:

• **Precision** = 100%

• **Recall** = 93.86% - 95.45%
TXPROBE - TESTNET TOPOLOGY

precision = 100%
recall = 97.40%
size → degree
color → Community unfolding

Higher community structure and modularity than random graph
CONCLUSIONS
The previous code was biased towards evicting transactions whose txid has a larger gap (lexicographically) with the previous txid in the orphan pool.
CONCLUSIONS

Select orphan transaction uniformly for eviction #14626

MarcoFalke merged 1 commit into bitcoin:master from sipp:201810_uniform_orphan_eviction 3 days ago

The previous code was biased towards evicting transactions whose txid has a larger gap (lexicographically) with the previous txid in the orphan pool.

randomize GETDATA(tx) request order and introduce bias toward outbound #14897

sipa merged 1 commit into bitcoin:master from naumenkog:master 10 days ago

This code makes executing two particular (and potentially other) attacks harder.

InvBlock
CONCLUSIONS

Select orphan transaction uniformly for eviction #14626

MarcoFalke merged 1 commit into bitcoin:master from sipa:2018_uniform_orphan_eviction 3 days ago

Conv Conversation 20   Commits 1   Checks 0   Files changed 1

sipa commented on 31 Oct 2018

The previous code was biased towards evicting transactions whose txid has a larger gap (lexicographically) with the previous txid in the orphan pool.

randomize GETDATA(tx) request order and introduce bias toward outbound #14897

sipa merged 1 commit into bitcoin:master from naumenkogs:master 10 days ago

Conv Conversation 115   Commits 1   Checks 0   Files changed 6

naumenkogs commented on 8 Dec 2018 • edited by MarcoFalke

This code makes executing two particular (and potentially other) attacks harder.

InvBlock
Is topology hiding a design goal, or is it a mean to achieve other goals (e.g: Transaction privacy)?
QUESTIONS
BONUS TRACK

Testnet vs Mainnet

INVBLOCKING (no-link)

Efficiency / Orphan pool eviction
WHY TESTNET AND NO MAINNET?

• TxProbe is rather invasive: it empties the MapOrphanTransactions pool of all nodes in the network every round

• We could not measure the implication that such behavior may have had on the propagation of regular transactions
INVBLOCKING V2

A's Mempool
∅

A

B

US

C's Mempool
∅

B's Mempool
∅
INVBLOCKING V2

A's Mempool
Ø
txF

US

B's Mempool
Ø
txP

C's Mempool
Ø
INVBLOCKING V2

A's Mempool
Ø

B's Mempool
Ø

C's Mempool
Ø

GETDATA (txF)

GETDATA (txP)
INVBLOCKING V2

A's Mempool
∅
txF

C's Mempool
∅

B's Mempool
∅
txP

A

C

B

US
INVBLOCKING V2

A's Mempool
∅

B's Mempool

C's Mempool
∅

A

B

C

US

txP

txF
INVBLOCKING V2
INVBLOCKING V2

A's Mempool
Ø txF

C's Mempool
txF

B's Mempool
txF

txF
INVBLOCKING V2

A's Mempool: Ø txF

B's Mempool: txP txF

C's Mempool: txF
INVBLOCKING V2
INVBLOCKING V2

A's Mempool
Ø txF

C's Mempool
txF

B's Mempool
txP

B

A

US
INVBLOCKING V2
INVBLOCKING V2

A's Mempool
Ø txF

C's Mempool

B's Mempool

US
INVBLOCKING V2

A's Mempool

C's Mempool

B's Mempool

A

B

US

C

txP

txF

(txP)

(txP)
INVBLOCKING V2

A's Mempool

B's Mempool

C's Mempool

$tx_P$

$tx_F$

$inblock$

$us$

$coinscope$

A

B

C
INVBLOCKING V2

A's Mempool

C's Mempool

B's Mempool

A

B

US

txF

txF

txF
INVBLOCKING V2

A's Mempool

B's Mempool

C's Mempool

txP

txF
INVBLOCKING V2

A's Mempool

A

B

C

C's Mempool

US
INVBLOCKING COMPARISON

A's Mempool

B's Mempool

US

C's Mempool

txP

txF

A

B

C

A

B

C

A

B

C

A's Mempool

B's Mempool

US

C's Mempool

txP

txF

A

B

C

A

B

C

A

B

C
MAKE THIS WORK IN A REAL NETWORK

Isolation

Synchrony

Efficiency

\[ \approx O(n) \]

\[ \approx O(\sqrt{n}) \]

\[ n = \#nodes \]
while (mapOrphanTransactions.size() > nMaxOrphans)
{
    // Evict a random orphan:
    uint256 randomhash = rng.rand256();
    std::map<uint256, COrphanTx>::iterator it = mapOrphanTransactions.lower_bound(randomhash);
    if (it == mapOrphanTransactions.end())
        it = mapOrphanTransactions.begin();
    EraseOrphanTx(it->first);
    ++nEvicted;
}

while (mapOrphanTransactions.size() > nMaxOrphans)
{
    // Evict a random orphan:
    uint256 randomhash = rng.rand256();
    std::map<uint256, COrphanTx>::iterator it = mapOrphanTransactions.lower_bound(randomhash);
    if (it == mapOrphanTransactions.end())
        it = mapOrphanTransactions.begin();
    EraseOrphanTx(it->first);
    ++nEvicted;
}


- Pick a random 256-bit value \( R \)
- Get the orphan transaction (O) with hash closer to, but greater than, \( R \)
- Evict O
- Repeat until mapOrphanTransaction is not full (default: 100)
- Double-spends are not checked for orphans
TXPROBE TRANSACTIONS OVERVIEW

UTXO_0

UTXO_1

Cleanser → Squatter_i

Flood

Parent_i → Marker_i
MAKE ROOM IN THE ORPHANPOOL

- Create the cleanser (regular transaction) and 100 squatters (double-spends between each other)

- Every squatter is created in a POW-ish way (e.g. re-sign until its hash falls bellow a certain threshold)

- All squatters are sent to the flood set nodes to replace any existing orphan.

- Finally, the cleanser is sent to empty the orphanpool
QUESTIONS