Part 1

Introduction
Problems with government-controlled money

- Unpredictable issuance
- Censorship and surveillance
- Political tool

“maintaining the dollar’s supremacy <...> is a critical strategic matter <...>. It is what allows us to have such effective sanction regimes around the world” – US Senator Tom Cotton (source)
A long way towards decentralized digital money

- eCash (David Chaum, 1982)
- Hashcash (Adam Back, 1997)
- bMoney (Wei Dai, 1998)
- RPOW (Hal Finney, 2004)
- Bit Gold (Nick Szabo, 2005)
- Bitcoin (Satoshi Nakamoto, 2008)

“Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending.”
Bitcoin in a nutshell

- Nodes broadcast transactions to a P2P network
- Miners produce blocks of transactions linked in a chain
- Each block proves that computational work has been performed to produce it
- Nodes choose the heaviest valid chain ⇒ consensus w/o trusted parties
- New coins enter circulation as miners’ rewards on a predictable schedule
Challenges for cryptocurrencies

- Privacy. Transactions are broadcast and stored in plaintext.
  Defenses against blockchain analysis (e.g., ZK) and network analysis.

- Scalability. All nodes validate all transactions.
  On-chain tweaks, alternative blockchains, and off-chain protocols (Lightning).

- Programmability. Bitcoin’s Script is (intentionally) limited.
  Alternative blockchains (Ethereum), smart contract programming languages.
Outline of this presentation

- Network-level privacy in Bitcoin and privacy-focused cryptocurrencies
  A well-connected adversary can cluster transactions issued from the same node.
- Security and privacy of the Lightning Network
  Privacy attacks on LN are likely if a few “important” nodes are compromised.
  Lightning’s throughput is limited for small payments.
  This limitation enables a new denial-of-service attack.
Part 2

Transaction clustering in Bitcoin and privacy-focused cryptocurrencies
Transaction propagation
Transaction propagation
Broadcast randomization
Our contributions

We show how a network adversary can link transactions issued from one node.

Key idea: transactions from the same issuer exhibit similar propagation patterns.

The plan:

1. Define the “transaction propagation pattern”
2. Quantify the “degree of similarity” between propagation patterns
3. Cluster transactions based on their propagation patterns
4. Measure the resulting decrease in anonymity
One transaction from Mallory’s perspective

<table>
<thead>
<tr>
<th>IP address</th>
<th>Received at (ms)</th>
<th>How likely to be “close” to the sender?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP₁</td>
<td>0</td>
<td>Highly likely</td>
</tr>
<tr>
<td>IP₂</td>
<td>10</td>
<td>Highly likely</td>
</tr>
<tr>
<td>IP₃</td>
<td>50</td>
<td>Likely</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>IP₁₀₀</td>
<td>5000</td>
<td>Highly unlikely</td>
</tr>
</tbody>
</table>
## Transaction propagation vector

For each transaction, first IPs to announce it are likely to be “close” to the sender.

We assign weights to IP addresses based on timestamps of announcements:

- Weight = 1 for the first
- Weight = 0 for (N+1)-th and all the following

<table>
<thead>
<tr>
<th>Tx</th>
<th>IP₁</th>
<th>IP₂</th>
<th>IP₃</th>
<th>IP₄</th>
<th>IP₅</th>
<th>IP₆</th>
<th>IP₇</th>
<th>...</th>
<th>IP₧</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0</td>
<td>t₂</td>
<td>t₃</td>
<td>t₄</td>
<td>t₅</td>
<td>t₆</td>
<td>t₇</td>
<td>...</td>
<td>tₘ</td>
</tr>
<tr>
<td>Weight</td>
<td>1</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>
Weight function

\[ w(t) = e^{(-t/k)^2} \]

where \( k \) is chosen such that \( w = 0.5 \) for the median timestamp:

Intuition: timestamp difference is more important if the timestamps are near zero.
Comparing weight vectors

<table>
<thead>
<tr>
<th>Tx</th>
<th>IP₁</th>
<th>IP₂</th>
<th>IP₃</th>
<th>IP₄</th>
<th>IP₅</th>
<th>IP₆</th>
<th>IP₇</th>
<th>IP₈</th>
<th>IP₉</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xa30e</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x35a6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x196c</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tx</th>
<th>0xa30e</th>
<th>0x35a6</th>
<th>0x196c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xa30e</td>
<td>1</td>
<td>-0.29</td>
<td>-0.45</td>
</tr>
<tr>
<td>0x35a6</td>
<td>-0.29</td>
<td>1</td>
<td>-0.43</td>
</tr>
<tr>
<td>0x196c</td>
<td>-0.45</td>
<td>-0.43</td>
<td>1</td>
</tr>
</tbody>
</table>
Correlation matrix clustering

With a right row-column permutation, clusters become visible.

Ground truth with our own transactions

We use our own transactions as ground truth:

- Issue ~40 transactions from two nodes
- Divide our transactions into “learning” and “control” sets
- Run the clustering algorithm assuming the knowledge of the “learning” set
- Assess the result based on how well the “control” transactions are clustered
Measuring anonymity

We use anonymity degree proposed* by Díaz et al.:

\[ d = -\sum_{i=1}^{N} p_i \log_2(p_i) / \log_2(N) \]

Where \( p_i \) is the estimated probability of the \( i \)-th tx to originate from the control set.

- \( d = 1 \): full anonymity
- \( d = 0 \): no anonymity

* Díaz, Seys, Claessens, Preneel. Towards measuring anonymity. 2002
Experiment outline

Putting all the pieces together:

1. Launch three well-connected, geographically distributed listening nodes
2. Log all transaction announcements, including the learning and control sets
3. Assign weights to vectors of IP addresses for each announcement
4. Calculate pairwise correlations between the weight vectors
5. Apply the spectral co-clustering algorithm to the correlation matrix
6. Calculate the anonymity degree using our own transactions

Let’s look at the results...
Bitcoin testnet. Anonymity degree = 0.63
Privacy-focused cryptocurrencies

Dash: Bitcoin Core fork
Monero: implemented from scratch
Zcash: Bitcoin Core fork

Various cryptographic and application-level techniques are used.

What about network-level privacy?
Zcash. Anonymity degree = 0.86
Monero and Dash

monero-mainnet
Assumed ~ 12 tx / cluster, 18 clusters. N = 3

dash-mainnet. N = 4, 9 clusters
Summary of part 2

P2P traffic reveals links between transactions from one sender.

Advice for users:

● Don’t issue multiple transactions from the same session
● Run nodes with an increased number of connections
● Periodically drop random connections and establish new ones

Advice for developers:

● Implement stronger broadcast randomization
● New P2P protocols: Dandelion and Erlay
  ○ Both prevent our attack: transactions are initially sent to outgoing connections only
Part 3

Security and privacy of the Lightning Network

Joint work with Pedro Moreno-Sanchez and Matteo Maffei (TU Wien)
Off-chain protocols

- Idea: let’s move (most of the) transactions *off-chain*
- Pros: high throughput, no changes to the main protocol
- Cons: new security and privacy challenges
Payment channel example

Off-chain

On-chain

(Alice, Bob): 10

Alice: 9 8 7

Bob: 1 2 3

✔: Alice, ? Bob

Alice: 7

Bob: 3

✔: Alice, ✔: Bob
Payment channel network

- Expensive to open channels between every two users (fees, confirmations)
- Solution: a network of payment channels
- Must ensure atomicity in multi-hop payments
Lightning Network architecture

- LN ensures atomicity with hash time-locked contracts (HTLCs)
- Coins go to Bob if he shows a hash preimage before time $t$, otherwise to Alice
Source routing

- Nodes gossip about channels available for routing
- Each node compiles a local view of the network
- The sender chooses the route based on the local view
- If a payment fails, the sender tries another route
Our contributions

What do LN’s security, privacy, and throughput depend upon?

In this work, we:

- quantify the effect of three previously described attacks*
- analyze a limitation on payment concurrency
- describe a new DoS attack vector

Value privacy

Attacker learns how much is being transacted.

Trivial for on-path adversaries: amounts are in plaintext.

Sufficient condition: 1 attacker’s node on the path.
Relationship anonymity

Attacker learns who pays whom (with probability much better than random guess)

Payments are linked by the same hash value.

Sufficient condition: 2 attacker’s nodes on the path: the first and the last.

(103 sat, h, t₃) (102 sat, h, t₂) (101 sat, h, t₁) (100 sat, h, t₀)
Wormhole attack

Attacker “shortcuts” a payment, taking fee from the honest node.

Damage for the honest node: a) no fees, b) capital locked until timeout expires.

Sufficient condition: 2 attacker’s nodes on the path with honest nodes in between.
How likely is an attack to succeed?

It depends on various factors:

- The type of the attack
- The payment amount
  - Smaller payments have more routing options
- How many nodes are compromised
- Which nodes are compromised

We aim to quantify the importance of these factors based on real LN data.
Experiment outline

- Assume that a certain subset of nodes is compromised
- Find all suitable paths between random sender and receiver
- Calculate the share of paths vulnerable to a given attack
- Average the result across many random runs

<table>
<thead>
<tr>
<th>Path</th>
<th>VP</th>
<th>RA</th>
<th>WA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Path 1</td>
<td>Safe</td>
<td>Safe</td>
<td>Safe</td>
</tr>
<tr>
<td>Path 2</td>
<td>Prone</td>
<td>Safe</td>
<td>Safe</td>
</tr>
<tr>
<td>Path 3</td>
<td>Prone</td>
<td>Prone</td>
<td>Safe</td>
</tr>
<tr>
<td>Path 4</td>
<td>Prone</td>
<td>Prone</td>
<td>Prone</td>
</tr>
<tr>
<td>Prone</td>
<td>75%</td>
<td>50%</td>
<td>25%</td>
</tr>
</tbody>
</table>
Results: highest degree nodes compromised
Results: + highest capacity nodes compromised
Results: + random nodes compromised
Trade-off between connectivity and privacy

- Routing via large nodes: dangerous if they are compromised
- Routing via small nodes: less liquidity and uptime
HTLC limit

How many concurrent payments can LN handle?

- One channel may hold multiple unresolved HTLCs
- Channel parties must be able to dispute malicious closures on-chain
- Dispute transactions include all unresolved HTLCs
- Bitcoin transactions must be < 100 KB
- Consequently, a channel supports at most 966 HTLCs (*HTLC limit*)
Example of HTLC depletion

Consider a channel with capacity of 1M sat.

967-th HTLC cannot be added, though the capacity is not depleted.

<table>
<thead>
<tr>
<th></th>
<th>Unresolved HTLCs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HTLC (to Alice, 1000 sat, 0xdf86...)</td>
</tr>
<tr>
<td>2</td>
<td>HTLC (to Bob, 1000 sat, 0x0a1f...)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>966</td>
<td>HTLC (to Alice, 1000 sat, 0x6f26...)</td>
</tr>
</tbody>
</table>

Total value of HTLCs (sat) 966k < 1M

Number of HTLCs 966
Limiting factors for throughput

Two limiting factors: channel capacity and HTLC limit.

Which one is more important? It depends on the amount:

- 0 – 546 sat (dust limit): no HTLC created
- 546 – **2700 sat** (0.3 USD): HTLC limit is more important
- >2700 sat: capacity is more important

We refer to 2700 sat as the **borderline amount**.
Evolution of borderline amount

Borderline amount relatively stable since 2019.
Up to 50% of channels affected

Nearly half of all channels could have handled more concurrent micropayments.
DoS by exceeding the HTLC limit

- An attacker blocks a channel by sending 966 near-dust payments
- Does not require as many coins as in the victim channel
- Can block one channel with $966 \times 546 = 527k$ sat (~60 USD)
  - Can block N times more more with an N-hop payment

<table>
<thead>
<tr>
<th>Channel capacity (sat)</th>
<th>Cost of depleting one channel (sat)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Capacity-based</td>
</tr>
<tr>
<td>100k</td>
<td>100k</td>
</tr>
<tr>
<td>1M</td>
<td>1M</td>
</tr>
<tr>
<td>10M</td>
<td>10M</td>
</tr>
</tbody>
</table>
Summary of part 3

- Privacy attacks are possible with only a few “important” nodes compromised
  - In September 2019, one entity (LNBIG) controlled 40% of LN capacity
- Throughput is constrained by the HTLC limit, in addition to channel capacity
- HTLC limit is relevant for micropayments (< 0.3 USD)
- A new DoS vector allows to block large channels cheaply
List of publications

4. Tikhomirov, Voskresenskaya, Ivanitskiy, Takhaviev, Marchenko, Aleksandrov. “SmartCheck: Static Analysis of Ethereum Smart Contracts”. WETSEB@ICSE 2018
Thank you! Questions?

Feel free to reach out:

- Twitter: @serg_tikhomirov
- Telegram: @s_tikhomirov
- sergey.s.tikhomirov@gmail.com
Image sources

- [https://explorer.acinq.co/](https://explorer.acinq.co/)
- [https://twitter.com/JustinAHorwitz/status/1248384733156741123/photo/1](https://twitter.com/JustinAHorwitz/status/1248384733156741123/photo/1)
- [https://www.reddit.com//fnchzr/](https://www.reddit.com//fnchzr/)
- [https://en.bitcoin.it/wiki/Promotional_graphics](https://en.bitcoin.it/wiki/Promotional_graphics)
- [https://twitter.com/ValueOfBitcoin/status/1258887539903148032](https://twitter.com/ValueOfBitcoin/status/1258887539903148032)