2019 IEEE European Symposium on Security and Privacy (EuroS&P)

Deanonymization and linkability of cryptocurrency transactions based on network analysis

Alex Biryukov
University of Luxembourg
alex.biryukov@uni.lu

Sergei Tikhomirov
University of Luxembourg
sergey.s.tikhomirov@gmail.com

***Abstract*—Bitcoin, introduced in 2008 and launched in 2009, is the first digital currency to solve the double spending problem without relying on a trusted third party. Bitcoin provides a way to transact without any trusted intermediary, but its privacy guarantees are questionable. Despite the fact that Bitcoin addresses are not linked to any identity, multiple deanonymization attacks have been proposed. Alternative cryptocurrencies such as Dash, Monero, and Zcash aim to provide stronger privacy by using sophisticated cryptographic techniques to obfuscate transaction data.**

Previous work in cryptocurrency privacy mostly focused on applying data mining algorithms to the transaction graph extracted from the blockchain. We focus on a less well researched vector for privacy attacks: network analysis. We argue that timings of transaction messages leak information about their origin, which can be exploited by a well connected adversarial node. For the first time, network level attacks on Bitcoin and the three major privacy-focused cryptocurrencies have been examined. We describe the message propagation mechanics and privacy guarantees in Bitcoin, Dash, Monero, and Zcash. We propose a novel technique for linking transactions based on transaction propagation analysis. We also unpack address advertisement messages (`ADDR`), which under certain assumptions may help in linking transaction clusters to IP addresses of nodes. We implement and evaluate our method, deanonymizing our own transactions in Bitcoin and Zcash with a high level of accuracy. We also show that our technique is applicable to Dash and Monero. We estimate the cost of a full-scale attack on the Bitcoin mainnet at hundreds of US dollars, feasible even for a low budget adversary.

I. Introduction

Bitcoin was, and still to some extent is, misleadingly referred to as an anonymous currency [42]. Indeed, unlike traditional financial systems, Bitcoin addresses are not tied to any real-world identity at the protocol level, but this fact alone does not guarantee strong privacy. Bitcoin transactions are broadcast through a peer-to-peer network in cleartext; after being verified by miners they are stored in a massively replicated shared database (the blockchain). A common technique to improve privacy in Bitcoin is to use a fresh address for every transaction (generating addresses is only limited by the size of the 256-bit key space). This piece of advice, often implemented in wallets, is no panacea, as the relationships between transactions can be inferred through blockchain analysis.

Multiple cryptographic techniques have been proposed to address the Bitcoin privacy problem, from services on top of the original protocols such as mixers to new alternative cryptocurrencies such as Dash, Monero, and Zcash. Dash relies on built-in background mixing powered by the so called masternode network. Monero implements ring signatures and confidential transactions. Zcash uses zero-knowledge proofs, namely, zk-SNARKs (though the majority of transactions do not take advantage of them due to heavy performance cost). Zcash and Dash are based on a fork of the Bitcoin Core codebase, while Monero is not.

Previous attacks on the privacy of cryptocurrency transactions mostly employed some form of data analysis on the transaction graph. We take another approach and analyze propagation times of protocol messages to infer relationships between transactions.

The ultimate goal of deanonymization is to reveal the relationship between cryptocurrency transactions (or addresses) and real-world identifiers, such as IP addresses. In our model, the goal of the adversary in our model is to infer a connection between a cryptocurrency transaction and the IP address of a node which was the first to introduce it into the network.[1] We rely on the core observation that a node can be uniquely identified by its set of connected peers (*entry nodes*). Earlier network-based deanonymization attacks [16] and [30] only took into account the first node to propagate a given transaction to the adversary. Our approach is more sophisticated. We apply carefully chosen weight functions to message timing information. This allows us to link transactions broadcast from one node, even if all addresses involved are unrelated (consequently, blockchain analysis would gain no insight).

Instead of associating transactions with IP addresses directly, we first cluster the transactions, and then try to assign IP addresses to clusters. Even if the latter step gains no insight, the clustering data used in combination with information from other sources is useful for the attacker. Moreover, our technique does not simply produce a binary decision (whether two transactions are related), but also allows for manual visual inspection of transaction clusters using heatmaps.

The rest of the paper is organized as follows. Section II provides an overview of the propagation mechanisms in various cryptocurrencies. Section III describes our approach to transaction clustering based on propagation timing. We implement and evaluate our technique on real-world cryptocurrencies. We were able to cluster our own transactions in Bitcoin and

[1]Even though an IP address is not linked to a physical person, it can be used to determine a relatively precise location of the device involved, and can be linked to a real-world identity if the responsible ISP is compromised.

DOI 10.1109/EuroSP.2019.00022

Zcash with high levels of precision and recall. In particular, in the case of Zcash, we can cluster transactions involving both transparent and shielded addresses. We also show that our technique is applicable to Dash and Monero. We provide rough calculations of the necessary resources and the monetary cost of an attack on the Bitcoin mainnet. We discuss attack scenarios for different types of wallets, and give a number of recommendations for users who want to preserve their privacy, as well as for developers of cryptocurrency protocols and wallets who want to give users an easier way to do so. Section IV provides an overview of related work, and Section V summarizes and suggests future work.

II. Background

A. Propagation of messages in cryptocurrency networks

Cryptocurrencies use P2P networks to disseminate messages. We now describe the relevant details on the networking behavior of Bitcoin (most alternative cryptocurrencies inherit these properties).

1) Address propagation: A newly launched node first performs a DNS lookup of a few records hard-coded into the software to discover the IP addresses of bootstrap nodes. It then asks the bootstrap nodes for (a subset of) the list of IP addresses of nodes known to them. Upon receiving the lists, the new node establishes a preconfigured number of connections with a random set of nodes, which we will refer to as *entry nodes*. If the TCP port 8333[2] is open, a node allows up to 117 incoming connections to be established (this number can be overridden in the configuration).

After joining the network and establishing connections, a node advertises its IP address (as seen from the Internet) in an `ADDR` message to its neighbors. Upon receiving an `ADDR` message, each node decides individually for each address whether to relay it to one or two of its neighbors, depending on reachability. A node re-advertises its address with random delays, every 24 hours on average. Nodes may also at any time query their neighbors for a list of addresses known to them (`GETADDR`); the response is an `ADDR` message containing up to 1000 addresses of peers recently seen on the network.

2) Transaction propagation: Propagation of transactions is a three step process. A node which has a new transaction advertises this fact to its neighbors with an `INV` (inventory) message containing the transaction hash only. Upon receiving an `INV`, each node decides whether to request the transaction content. If the node does not yet have the transaction, it replies with a `GETDATA` message and receives the transaction contents in a `TX` message. Blocks are propagated in a similar manner.

3) Randomization: A straightforward way to broadcast messages in a P2P network is to relay them as soon as possible to all neighboring peers. Recognizing that this approach may harm privacy, Bitcoin developers introduced randomness in this process. Based on the related work and the source code of the major cryptocurrencies, we distinguish three propagation mechanisms:

- Naïve gossip: broadcast to all neighbors as soon as possible (used in Monero);
- Trickling: for a number of fixed-length time periods, broadcast to a new random subset of neighbors (used in Zcash and Bitcoin pre-2015);
- Diffusion: broadcast to each neighbor after a random delay (used in Dash and Bitcoin post-2015).

[2]The default port for the Bitcoin mainnet. Other networks use other ports by default: 18333 for the Bitcoin testnet, 8233 for Zcash, 18080 for Monero, 9999 for Dash.

B. Alternative cryptocurrencies

We now provide a brief description of the three privacy focused cryptocurrencies we consider.

1) Zcash: Zcash [6] implements the Zerocash protocol [14] [27] – an improvement of an earlier Zerocoin protocol [34]. It uses zk-SNARKs [15] to hide the transaction information, while still allowing anyone to verify its correctness. Zcash does not provide privacy by default as of late 2018: zk-SNARKs are used only in a small minority of transactions involving *shielded* addresses [29]. The majority of transactions happen between *transparent* addresses and have no additional privacy-preserving mechanisms compared to Bitcoin.

Zcash codebase was forked off Bitcoin core in November 2015 at version 0.11.2 (commit 7e27892). In 2015, Bitcoin changed the network propagation mechanism from trickling to diffusion [53] (commit included in version 0.12). According to [26], this provided only marginal privacy improvements. Zcash did not port those modifications and still uses trickling.

In October 2018, Zcash underwent an update code-named Sapling [54], which greatly increased performance of shielded transactions. This allows for shielded transactions to be supported in light wallets, including mobile ones.

2) Dash: A distinguishing feature of Dash [4] is a two-tier architecture. Along with regular nodes, users may set up so-called *masternodes*, which require a 1000 DASH collateral (approximately 160 000 USD, at the time of writing). The Dash network contained around 5000 masternodes in late 2018. Masternodes [10] receive 45% of the mining reward for providing additional services:

- *PrivateSend* – a privacy-enhancing transaction type, where a random masternode deterministically chosen based on the latest block hash matches users who wish to mix their coins;
- *InstantSend* – a technique to increase merchants' confidence in accepting transactions without waiting for them to be included in a block, where a random subset of ten masternodes agrees on the "locked" set of inputs.

The Dash networking protocol is based on Bitcoin's but substantially more complex: in addition to Bitcoin message types, it contains 22 new ones related to masternode functionality [5]. Dash uses the diffusion mechanism ported from Bitcoin.

3) Monero: Monero [9] implements the CryptoNote protocol [50]. Monero is not based on the Bitcoin Core codebase.

The Monero community recognizes the threat of deanonymization through network analysis [45][31][24][19]. The developers are integrating an I2P router into Monero (the Kovri project [7]), but it is not yet deployed as of November 2018. Monero does not have any broadcast randomization such as trickling or diffusion.[3]Further inspection of the source code and an answer on a Monero-related Q&A site reveals that Monero nodes do not limit the number of incoming connections by default [48]. Monero is the only one of the three privacy-preserving currencies which is private by default: users do not have to explicitly choose the "private" option (such as a shielded address in Zcash and PrivateSend in Dash). In October 2018, Monero version 0.13.0 introduced an implementation of Bulletproofs – a cryptographic technique which allowed greatly reduced transaction size (and hence fees), which also, similar to Sapling in Zcash, is expected to incentivize Monero adoption on devices with limited resources [47].

III. Our approach

A. Our approach

1) Intuition: Our goal is to cluster transactions based on the node which was the first to introduce them into the network. Consider the first N nodes which relayed a transaction to our listening node. We assign weights to IP addresses of nodes depending on the propagation timestamps. Intuitively, a peer that relays a new transaction to us quickly is likely to be an entry node or closely connected to one. Our clustering algorithm is based on the weight vectors of transactions. We expect transactions originating from one node to yield relatively well-correlated weight vectors.

Due to broadcast randomization, we do not expect all transactions from one node to be well-correlated. But the matrix of pairwise correlations exhibits special behavior which would help us infer transactions clusters nevertheless. Consider a node with eight entry nodes with IP addresses (p_1 to p_8) making three transactions: tx_1, tx_2, tx_3. If transactions were broadcast in batch via the same subset of the entry nodes, their weight vectors would be very similar. But due to diffusion or trickling, the following scenario is more typical: tx_1 quickly relayed from $p_{\{1,2,3\}}$, tx_2 from $p_{\{3,4,5\}}$, tx_3 from $p_{\{5,6,7\}}$. If we considered only the first propagation, these transactions would seem completely unrelated. But with weight vectors, considering that those are sparse, the correlation between tx_1 and tx_2 and between tx_2 and tx_3 would be noticeable, which would allow us to reveal not only the relationship between these pairs but also among all three transactions. Note that this technique is also applicable for transactions originating from a light client (in this case, a cluster represents transactions from multiple clients connected to the same full node).

2) Data collection and representation: We use a modified Bitcoin network probing tool `bcclient` [40] to maintain parallel connections to peers and log incoming messages:

[3]See `relay_notify_to_all` at https://github.com/monero-project/monero/blob/master/src/p2p/net_node.inl#L1515.

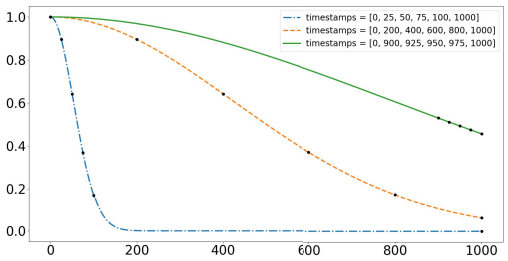

Fig. 1: Weight functions for three timestamp vectors

transaction hash, the IP which relayed it to us, and the timestamp of this event.

We use Python scripts to extract the essential information from the log, save it in a more compact JSON format, analyze the data, and visualize the results. For each transaction, we save a list of *(t, IP)* pairs, where t is a *relative* timestamp (i.e., we subtract the timestamp of the first propagation of this transaction from all its propagations).

3) Weight functions and clustering: Let tx be a transaction. Let $p^{tx} = [p_1^{tx}, p_2^{tx}, ...p_N^{tx}]$ be the vector of the first N IP addresses which relayed tx to us. Let $t^{tx} = [t_1^{tx}, t_2^{tx}, ...t_N^{tx}]$ be the vector of the corresponding relative timestamps. For each $p_i^{tx} \in p^{tx}$, we assign a parameterized weight as follows:

$$w_k(p_i^{tx}) = e^{-(t_i^{tx}/k)^2}$$

The weight function is chosen to reflect the decreasing importance of every next broadcast. p_1 is assigned the maximal weight of 1.0 (note that $t_1 = 0$ by definition); other nodes receive lower weights. Our experiments show that this function family yields better clustering (compared to $1/(kt)$ and e^{-kt}). The intuition is that it gives higher weights to a certain window depending on k while exponentially decreasing outside of it. Moreover, window size is adjusted for each vector.

For each p^{tx}, we want to use such w_k that gives sufficient variance among the weight values. Weights quickly fall to nearly zero if k is too low and stay close to one if k is high. Let t_{med}^{tx} be the median value in t^{tx} (average of the high and low medians if the length of t^{tx} is even). We choose k_{opt}^{tx} s.t. the weight of t_{med}^{tx} would be 0.5:

$$k_{opt}^{tx} = \frac{t_{med}^{tx}}{\sqrt{-\ln(0.5)}}$$

This choice of k distributes the weights for any t^{tx}: they neither stay close to one nor quickly fall to zero (see examples in Figure 1). For each transaction, we evaluate the vector of weights:

$$w^{tx} = w_{k_{opt}^{tx}}(t^{tx})$$

Let X be the set of all transactions we consider. Let P be the set of IP addresses of nodes which appeared in at least one of p vectors in X:

$$P = \bigcup_{tx \in X} p^{tx}$$

We define an extended weight vector v_{tx} for each tx by setting the weight of nodes in $P \backslash p^{tx}$ to zero and sort the values in the weight vectors w. r. t. the alphabetical order of P. We then calculate a matrix where an element in i-th row and j-th column is the Pearson correlation of the extended weight vectors v_{tx_i} and v_{tx_j}. This matrix can supposedly be transformed into a block-diagonal matrix with blocks (clusters) corresponding to transaction sources.

To reveal the clusters, we use spectral co-clustering [21] implemented in the Python `sklearn.cluster.bicluster` module [46]. Given an input matrix A, the algorithm preprocesses it as follows:

$$A_n = R^{1/2} A C^{-1/2}$$

Where R is the diagonal matrix with entry i equal to $\sum_j A_{ij}$, and C is the diagonal matrix with entry j equal to $\sum_i A_{ij}$.

The singular value decomposition of A provides the partitions of rows and columns: $A_n = U \Sigma V^T$. The $l = \lceil \log_2 k \rceil$ singular vectors provide the partitioning information. Let U be a matrix with columns $u_2, \ldots, u_{l+1}$, and similarly for V. Then Z is defined as:

$$Z = \begin{bmatrix} R^{-1/2} & U \\ C^{-1/2} & V \end{bmatrix}$$

The rows of Z are clustered using the k-means algorithm.

B. Quality assessment

1) Measuring clustering quality: We use the Rand score as an external metric of clustering quality, as described in [12] (Section 4.2). The Rand score operates on *pairs* of elements and reflects the proportion of "right decisions" regarding whether to put a pair of transactions into one or different clusters.

SS, SD, DS, and DD are numbers of transaction pairs defined as follows:

- SS: same cluster, same category (two of our transactions in the same cluster);[4]
- SD: same cluster, different category (our and foreign transactions in the same cluster);
- DS: different cluster, same category (two of our transactions in different clusters);
- DD: different cluster, different category (our and foreign transactions in different clusters).

Note that this assessment only considers clusters with "our" transactions, because we do not know whether any two "foreign" transactions should have been assigned to the same cluster:

$$R = \frac{SS + DD}{SS + SD + DS + DD}$$

[4]In our case, there are only two categories: "our" and "foreign" transactions.

We further modify this metric by parameterizing it with the minimal number of our transactions in a cluster required to consider it in the calculation. In our experiments, we only consider clusters with at least two of our transactions. With no such threshold, large clusters with one of our transactions disproportionately increase DD and bring the score close to 1.0, which does not reflect the subjective amount of information an adversary acquires.

2) Measuring the degree of deanonymization: To estimate the success rate of the attack, we use a quality score based on the *anonymity degree* proposed by Díaz et al. [22]. The anonymity degree is designed to measure the amount of information an attacker gains compared to perfect anonymity (where each user has an equal probability of being the originator of a given message). Let p_i be the probability that a transaction i originates from a given source $S_{control}$; N is the total number of transactions. The entropy is calculated as:

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i)$$

The maximal entropy is:

$$H_M = log_2(N)$$

The anonymity degree is defined as:

$$d = \frac{H(X)}{H_M}$$

The anonymity degree does not reflect the fact that the probability distribution obtained by the adversary may not be well aligned with the true probability distribution. To address this issue, we propose an *adjusted* anonymity degree. First, we calculate the median square error e between our probability distribution and the known true distribution (1 for transactions from $S_{control}$ and 0 for others), based on a subset of transactions from the control set. The adjusted anonymity degree is defined as follows:

$$d_{adj} = 1 - (1 - e) * (1 - d)$$

To explain on two edge case examples: If $e = 0$ (the attacker precisely predicted the distribution), $d_{adj} = d$; if $e = 1$ (the attacker's distribution does not at all reflect the reality), $d_{adj} = 1$ (the system retains full anonymity).

The assumptions of our model have their limitations. Our clustering technique depends on a user issuing a series of transactions in a relatively short time window of several minutes (up to an hour), through the same set of entry nodes (i.e. from the same session). If a user re-launches the client, their transactions issued before and after this event would not be linkable by our technique.

C. Experiment overview

Assume our goal is to cluster transactions originating from one target source $S_{control}$. We capture N transactions and know that n of them were issued from $S_{control}$; k of them are known to us. For each transaction i, we assign an a priori

probability of having originated from $S_{control}$: $p_i = n/N$. The outline of our experiment is as follows.

1) collect a fresh list of live network peers;
2) establish a number of parallel connections to them, log the timestamps of receiving `INV` and `ADDR` messages;
3) launch two nodes S_{learn} and $S_{control}$, so that their initial `ADDR` advertisements would be logged;
4) issue two series of transactions (the learning and the control sets) from S_{learn} and $S_{control}$ respectively;
5) for each considered number of first propagations, calculate the transaction correlation matrix;
6) run the clustering algorithm with various assumed average number of transactions per cluster;
7) choose the best clustering by Rand score on the "learning" set;
8) in the best clustering, assign the cluster weights proportionally to the distribution of k known transactions from $S_{control}$;
9) assign zero probability of being in $S_{control}$ to transactions from S_{learn};
10) re-distribute the probability weight among transactions in each cluster;
11) calculate the final adjusted anonymity score;
12) re-arrange the clusters such that large correlations would be close to the diagonal (closely correlated clusters should be close in the picture);
13) visualize the results as a heatmap.

D. Visualization

We use heatmaps to visualize the results. A heatmap is a matrix where each row and each column represents a transaction that we captured during an experiment. The color of the square at the intersection of the k-th row and n-th column represent the correlation of weight vectors of k-th and n-th transactions (darker is higher). Note the the heatmap is diagonally symmetric by definition (black squares along the main diagonal reflect the fact that a transaction has a 1.0 correlation with itself).

Our assumption is that there exist a permutation of rows and columns such that the highly correlated elements would be close to the main diagonal and would exhibit a block-diagonal structure, revealing possible relations between transactions issued from the same node. In the figures below, ticks along the axes indicate our transaction from the control set.

E. Evaluation

We evaluate our method by clustering our own transactions in Bitcoin (testnet and mainnet) and Zcash. For these experiments, we log the traffic (both `INV` and `ADDR` messages) for 15 minutes. The anonymity degree calculated on our own transactions indicates a substantial loss of privacy. For Dash and Monero, we ran the clustering algorithm without calculating an external quality metric. We obtained clearly visible clusters, which indicated that our approach is applicable for these cryptocurrencies as well.

1) Bitcoin testnet: We performed four experiments on the Bitcoin testnet. For all experiments, our listening node attempted to occupy up to 117 slots for all servers. We performed three independent experiments for three listener locations and a fourth experiment where we use all three servers simultaneously: we divide the fresh list of live peers into three equal parts, distribute them among the three servers, and then merge the three log files. The goal of the fourth experiment was to measure the advantage an adversary may gain from using geographically distributed servers. As listening nodes, we used Amazon EC2 servers in three geographical locations: Frankfurt (Germany), Tokyo (Japan), and North California (the US). In all experiments, test transactions were issued from computers located in Europe.

We issued two sets of test transactions (the learning and the control sets) containing 30 transactions each. We denote 10 transactions out of the control set as "known" to estimate the anonymity degree.

Note that the number of live peers collected by each of the listeners is very close, which indicates that we do obtain a complete view of the network. Note also that the number of received transactions varies little between experiments, whereas the number of `ADDR` messages is significantly higher in the experiment with three listeners. This confirms our hypothesis that address advertisements propagate through the network more slowly than transactions and blocks. The number of average available slots is independent of the location of the listener. The anonymity degree is lower (i.e., better for the attacker) in the Frankfurt experiment, which may be explained by the fact that the nodes issuing test transactions were closer to the listening nodes than in the other experiments. The joint experiments which combined information from three geographically distributed listeners gained the best results with an anonymity degree of 0.63.

a) Estimating the original IP: We use the ADDR messages to determine (with some level of precision) the IP of the node which issued the transactions in the control group. In our experiments, we first launch the listener, and only then launch the issuing nodes. This means that the listener captures the ADDR messages issued by the issuing nodes during bootstrapping. Address messages propagate through the network more slowly than transactions, which are relayed to all nodes' neighbors with relatively small random delays. A node listening to network traffic can therefore distinguish between messages containing addresses of recently joined nodes and re-broadcasts of older addresses messages. If an IP is relayed from 1 or 2 nodes, which is most often the case, we assume these IP addresses are old re-broadcasts. If an IP is relayed from a higher number of nodes, we assume the node at the relayed IP either has just joined the network or is re-advertising its IP after an approximately 24 hours delay.

Our idea is to leverage the `ADDR` messages as follows. For each cluster, we determine the IPs of the most "important" nodes, i.e., nodes we assume are entry nodes of the transaction originator. For each transaction in the cluster, we sum up the weights of all IPs which relayed it to us. The top 10% of most

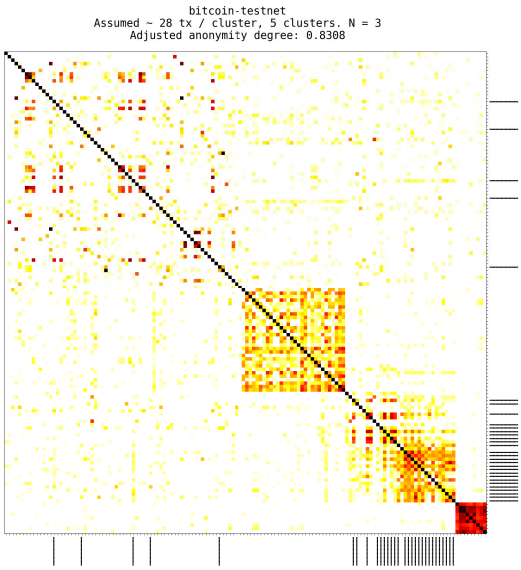

Fig. 2: Bitcoin testnet (California)

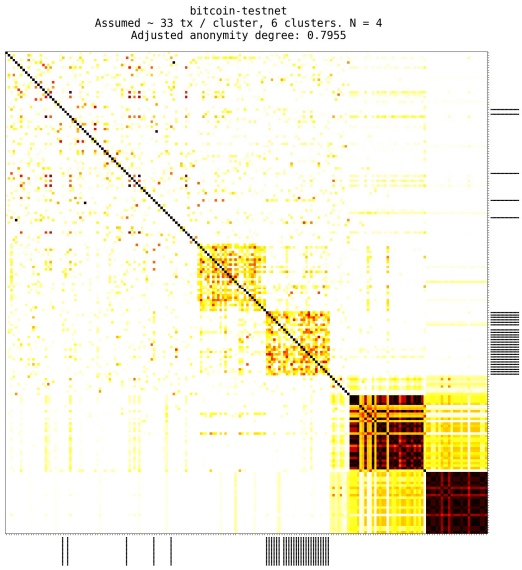

Fig. 3: Bitcoin testnet (Tokyo)

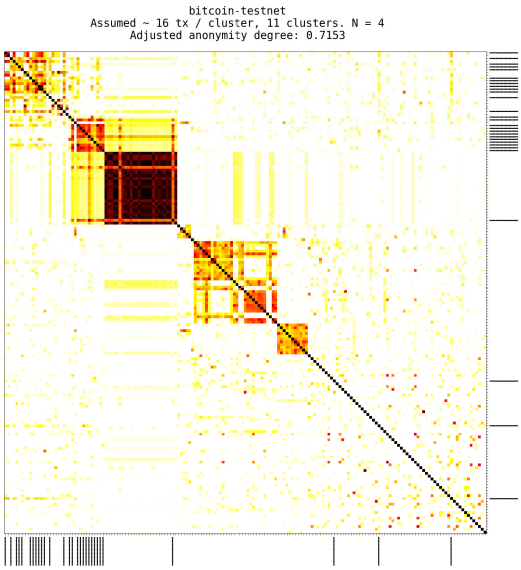

Fig. 4: Bitcoin testnet (Frankfurt)

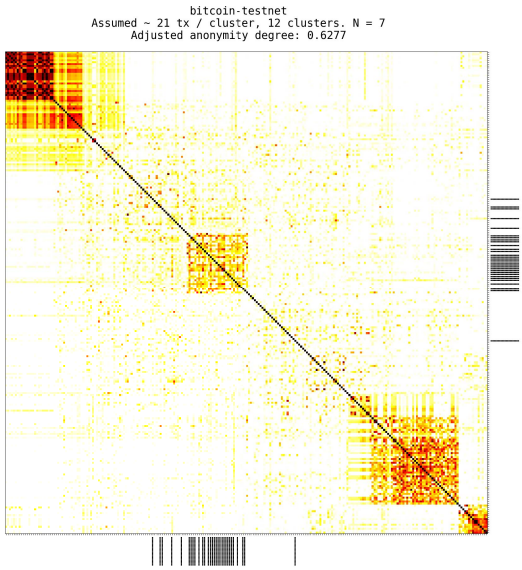

Fig. 5: Bitcoin testnet (combined)

TABLE I: Experiments on Bitcoin testnet and Zcash. The * sign indicates results obtained in experiments where we only connected to a subset of network nodes.

Network	Listener location	Anonymity degree	Servers	Avg free slots	Tx INVs	ADDRs
Bitcoin testnet	California	0.83	1141	64	139	402
Bitcoin testnet	Tokyo	0.80	1128	64	193	414
Bitcoin testnet	Frankfurt	0.72	1137	64	172	403
Bitcoin testnet	combined	0.63	1154	63	250	1321
Bitcoin mainnet	Frankfurt	0.88	1000*	25*	3238	11300
Zcash	Frankfurt	0.86	206	36	62	1086

weighted IPs are assumed to be the entry nodes. Looking at propagation of ADDR messages, the intuition is that an ADDR message relayed by a set of IPs which substantially intersects with assumed entry nodes of cluster X is the IP address of the node which issued the transactions. We define an IP to "likely" correspond to the transactions originator if it was relayed to us in an ADDR message by a set of IPs which overlap with assumed entry nodes determined on the previous step. Applying this technique to the Bitcoin testnet experiments, in 3 experiments out of 4 the right IP appeared in the top 5 most likely originator IPs for clusters which consist mostly of control transactions. This result indicates that in addition to being able to estimate with high accuracy whether given two transactions were issued by the same node, an adversary can narrow down the search for the IP of the transaction source to a handful of IPs. This may give the attacker valuable information, including the approximate geographical location of the victim.

2) Bitcoin mainnet: For Bitcoin mainnet, we performed one experiment with a listener located in Frankfurt. An experiment on Bitcoin mainnet showed that transactions also exhibit the "clustering" behavior, though the results are weaker because of a much larger transaction rate and due to the fact that we only connected to 1000 servers (asking for up to 50 connections). We used learning and control transaction sets of 20 transactions each; 5 transactions from the control set were assumed "known" for anonymity degree calculation.

3) Zcash: For the Zcash mainnet, we performed one experiment with a listener located in Frankfurt. The learning and control sets consist of 20 and 18 transactions respectively; 8 out of 18 control transactions are shielded (from a t-address to a z-address). We use 6 control transactions as "known" for anonymity degree estimation. The Zcash network is much smaller than the Bitcoin testnet. Moreover, Zcash servers have far fewer free slots on average (36 against 64 on Bitcoin testnet). We notice that relatively many servers only provide our listener with 1 – 10 slots. This may indicate a larger share of "protected" nodes, i.e., nodes which are configured (using firewalls or other network-level means) to only provide a limited number of connections to each IP. Note than a resourceful adversary may overcome this limitation by purchasing additional IP addresses from a cloud provider.

Note that our attack does does not take into account transaction content or type. Consequently, our method applied for Zcash allows clustering transactions involving both transparent and shielded addresses (transactions from the control set which involve shielded addresses are marked with longer ticks in the figure).

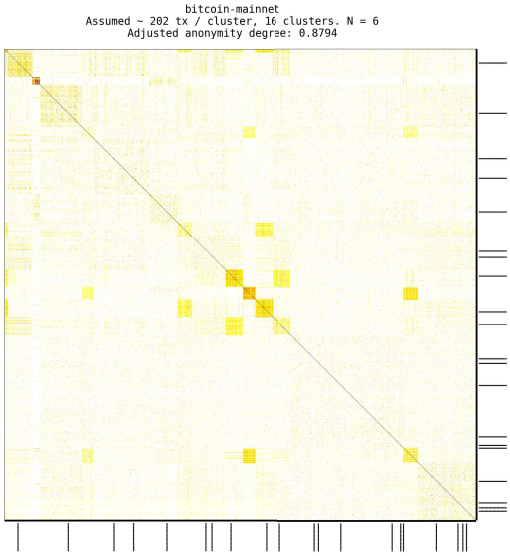

Fig. 6: Bitcoin mainnet

4) Dash: We ran an experiment on the Dash mainnet (connecting to 500 random nodes from the total of 3065, asking for 30 slots). In addition to announcing transactions and blocks, Dash uses the inventory mechanism for managing the masternode network [1], which includes periodic pings of masternodes to check whether they are functioning, managing mixing transactions, voting for governance proposals, etc. Our tool is not yet adapted for handling Dash-specific messages. The logs show many Dash-specific inventory messages, which do not later appear on block explorers (i.e., are not usual transactions). In a 15-minute experiment we received 12 transaction inventory messages and 396 Dash-specific messages. We ran our clustering algorithm two times: taking Dash-specific messages into account (Figure 10), and considering only usual transaction inventory messages (Figure 11).

In both cases, we obtained clearly visible clusters. These preliminary results demonstrate a clear privacy concern, espe-

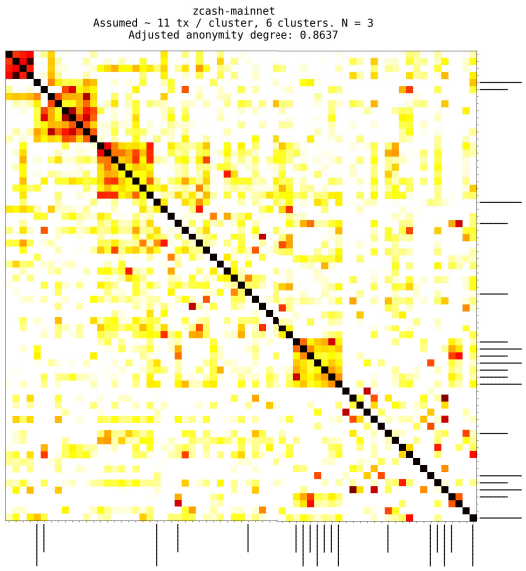

Fig. 7: Zcash

cially if network analysis is combined with deanonymization attacks on Dash based on transaction graph analysis [28].

5) Monero: Contrary to Bitcoin, that allows spending a transaction output before it is confirmed, Monero imposes certain restrictions. A new output appears as "locked" until the corresponding transaction gets 10 confirmations (20 minutes at the target block time of 2 minutes) [23]. Though this is a wallet-level restriction and not a protocol-level one, the major implementations (the official desktop wallet and Monerujo wallet for Android) support it. This means that the scenario of our previous experiments is rather unrealistic: for example, in order to issue 20 transactions within a 20 minute period, a user must have 20 independent, "unlocked" transaction outputs (each of which takes 20 minutes to create).

We check the suitability of our technique on Monero by performing an experiment without our own transactions. The goal of the experiment is to check whether we observe a block-diagonal structure of the correlation matrix between transaction propagation vectors.

We connected to 200 nodes (with a single connection per node) and received 124 transactions in a 38 minute window (see Figure 12).

The less clear picture compared to e.g. Bitcoin testnet may be explained as follows:

- We establish connections with too few nodes (200, while the total number of nodes is estimated at 1700-1800 [8]);
- The propagation mechanism is not optimized: there is no `INV` – `GETDATA` – `TX` exchange, transactions are relayed unconditionally to all neighbors, which blurs the picture. On the other hand, an adversary connecting to nearly all nodes is expected to gain a near-perfect insight into the original broadcasters of transactions, as IP addresses of transaction authors will most likely be among the first to relay a transaction to the adversary.

Another peculiarity which makes our analysis more difficult is that `monerod` connects to nodes relatively slowly (compared to `bcclient`). In our experiment, while trying to connect to 200 nodes, we got 150 connections only after approximately 2 hours, 175 connections after 3 hours, 200 connections after nearly 8 hours. We also notice that none of the hard-coded DNS seeds resolves (as of mid-July 2018); the client falls back to seed IP addresses (also hard-coded).

F. Estimation of costs for an attacker

We now estimate the resources required for a full-scale attack on the Bitcoin mainnet. As of November 2018, the Bitcoin mainnet consists of approximately 10 000 nodes. According to our measurements, the average number of free slots is 43 (measured on 1000 random peers). According to the Bitcoin protocol documentation [2], the size of an INV message is "36x + const for message with x objects". We assume an INV for a single transaction requires 40 bytes. An average Bitcoin transaction rate, as of November 2018, is around 250 000 transactions per day, or 2.89 tx/s. Assuming each connection eventually relays each transaction, we arrive at the required bandwidth for one connection slot as: 2.89 tx/s * 40 b/tx = 115.6 b/s. A full-scale attack on Bitcoin mainnet would require maintaining an average of 43 connections to 10 000 nodes, i.e., a total bandwidth of 115.6 b/s * 10000 nodes * 43 slots/node = 49708000 b/s = 47.4 Mb/s = 379 Mbit/s. An hour-long attack at this bandwidth will require receiving approximately 167 GB of incoming traffic.

We may estimate the monetary cost of the attack based on the costs of running a Bitcoin full node on a cloud server. Various estimations put that cost at between $3 and $20 per month [55][20] Bitcoin Core maintains 8 outgoing and accepts up to 117 incoming connections by default. In our measurements, an average Bitcoin server has 43 open slots. Assuming it has a total of 125 slots, 125 - 43 = 82 slots eventually get occupied. An adversary needs to maintain 10 000 * 43 = 430 000 connections, or 5244.0 times the bandwidth of a regular node. A 30-days month is 720 hours. Considering all of the above, we conclude that an estimated cost of an hour-long attack is approximately 5244 / 720 = 7.3 times the monthly cost of running a full node. That leads to an estimation of bandwidth costs at $20 – 150. Even taking into account the cost of computation and storage, the total cost of the attack is on the order of hundreds of US dollars – well within reach of even amateur adversaries, not to mention professional black-hat hackers and nation states.

All our experiments on Bitcoin testnet and Zcash mainnet cost $35 (this can probably be decreased by optimizing the scripts, immediately copying the data to a local machine and deleting it from the cloud, etc).

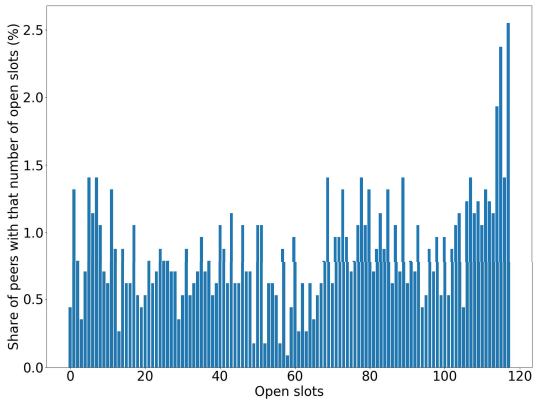

Fig. 8: Free slots: Bitcoin testnet

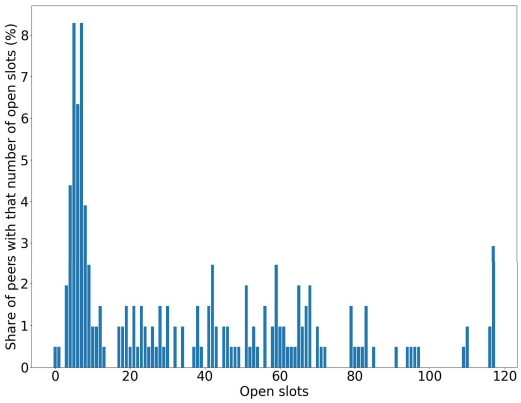

Fig. 9: Free slots: Zcash mainnet

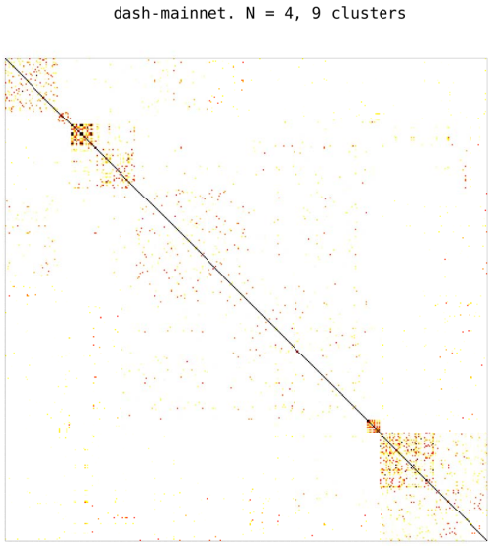

Fig. 10: Dash (Dash-specific messages and usual transactions)

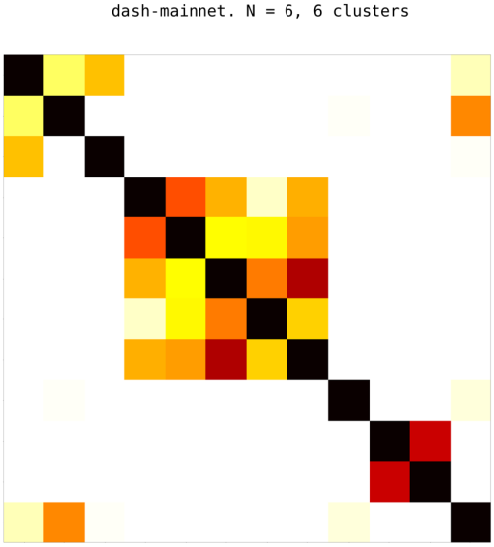

Fig. 11: Dash (usual transactions)

G. Ethical considerations

All linkage experiments were done on our own transactions and when possible on the testnets. The experiments on the Bitcoin mainnet deliberately did not attempt to occupy all connection slots, and operated only on a subset of 1000 nodes (out of approximately 10 000). Logs from mainnet experiments will be deleted.

H. Discussion and mitigations

1) Attack scenarios and countermeasures: We now discuss the possible attack scenarios and countermeasures.

Application-level cryptographic techniques, such as zero-knowledge proofs in Zcash, can not defend against our attack, as we only consider transaction hashes and their propagation times, ignoring their content.

A popular mitigation for deanonymization attacks based on network analysis is to use anonymity overlay networks such as Tor [11], or mix networks such as Loopix [39]. In our case, this countermeasure is inefficient: transactions issued by the same cryptocurrency node can be linked by a global passive adversary even if the data was transferred through Tor or other anonymity network before being publicly broadcast.

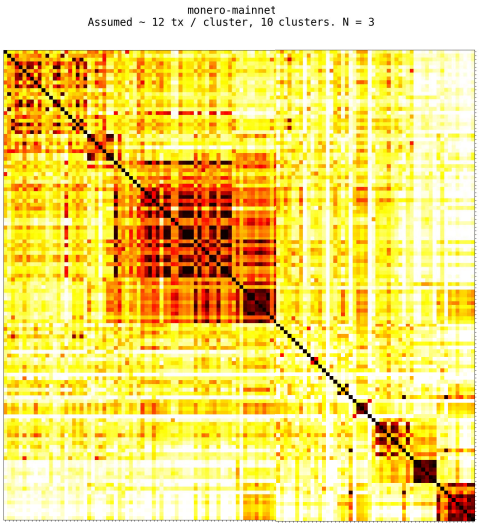

Fig. 12: Monero

Tor helps to hide the relationship between IP addresses of the originating node and the first node to broadcast the transaction to the peer-to-peer network, but we cluster transactions based on the first broadcaster's entry nodes (or nodes topologically close to those in terms of network propagation times), not the IP addresses of the originating node. Note that broadcasting transactions via Tor may even introduce additional man-in-the-middle vulnerabilities [17] (the situation is similar to the case of a light wallet described below).

We distinguish three cases depending on the type of the user's wallet.

a) Full node with incoming connections (server): A typical operator of a Bitcoin server is either a business (wallet provider, exchange, etc), or an enthusiast willing to donate their computing resources to help the network. In the first case, the transaction relayed through the node may originate from multiple users of this business, which also harms their privacy. The full node operator may implement the following countermeasures:

- Run the node with an increased number of outgoing connections to dilute the quality of the topological fingerprint;
- Use additional random delays on top of those implemented in the node software;
- Drop connections to randomly chosen entry nodes and establish new ones, constantly altering the set of entry nodes;
- Give advice to users not to broadcast sensitive transactions within a short period of time.

b) Full node without incoming connections: Transactions originating from a full node without incoming connections (ex. computer behind NAT) may be clustered based on the set of entry nodes. In order to prevent that, the user can re-launch the software after making a transaction, so that each transaction would be broadcast through a new set of entry nodes.

c) Light wallets: The majority of Bitcoin users use light wallets, i.e., they delegate validation to another full node using simple payment verification (SPV). From the networking perspective, most light wallets, especially mobile ones, do not even connect to a P2P network. Instead, they send transactions to the server of the wallet provider via TLS, which in turn broadcasts them to the P2P network.[5] Proposed countermeasures for light wallets would be:

- Use wallets that connect to the actual P2P network and broadcast transactions without relying on a centralized server (e.g., Bitcoin wallet for Android [3]);
- Use different light wallets for transactions not meant to be linkable;
- If the above advice is inapplicable, at least choose a popular light wallet to increase the anonymity set.

2) Recommendations for core developers: Cryptocurrency developers should introduce privacy enhancing measures at the network level, especially if the currency is meant to be privacy-preserving. As our results show, trickling and diffusion, as they are implemented in Bitcoin and its forks, are not sufficient. A promising proposal for anonymous peer-to-peer broadcast is Dandelion [51][25] (see Section IV for an overview).

IV. Related work

A. Privacy in cryptocurrencies

Most early research on security and privacy of cryptocurrencies only considered Bitcoin as the dominant cryptocurrency at that time and was primarily focused on blockchain analysis [33][38][43]. Reid et al. [42] and Androulaki et al. [13] provide an overview of privacy challenges in Bitcoin. A popular mitigation, which does not require modifications to the Bitcoin protocol, is mixing. A Bitcoin transaction spends a number of unspent transaction outputs (UTXO) as inputs and generates a number of new UTXOs. Mixing allows users to create a joint transaction that combines all relevant inputs and outputs, making it harder for an adversary to track the flow of coins of a single user. The major drawback of this scheme is that users must agree to co-sign the transaction using additional means of communication. This process is unscalable without coordination by a trusted third party. Bonneau et al. [18] propose Mixcoin, a protocol to automate mixed payments in Bitcoin and similar cryptocurrencies which includes an accountability mechanism to expose theft. Valenta et al. [49] add a blind signature scheme to Mixcoin to prevent the operator from spying on users. Alternative implementations of mixing protocols include CoinJoin [32] and CoinShuffle [44].

[5]Apart from clustering, this poses an arguably more serious privacy threat, which is outside the scope of this work: the wallet provider can log all users' transactions and link them to their IP addresses. Using Tor is not applicable in this case, as the wallet servers will still be able to associate a user's transactions by other means (e.g., by making the wallet send a cookie along with transactions).

Quesnelle [41] proposes a method to link Zcash transaction based on a heuristic extracted from real-world usage of transparent and shielded addresses.

B. Network analysis

Koshy et al. [30] analyze Bitcoin's anonymity through the lens of P2P network properties. They propose a technique for a global passive adversary to deanonymize users based on transaction propagation times. The adversary aggregates network traffic into tuples containing the Bitcoin address, the first IP address to relay this transaction, and the transaction identifier. For each transaction, the tuples are constructed for each input and output. Each tuple is counted as a "vote" in favor of a hypothesis that a certain IP "owns" (i.e., possesses the private key of) a certain Bitcoin address. While this paper provided valuable insights, it seems not to account for trickling / diffusion, which must have decreased the quality of the proposed deanonymization algorithm.

Biryukov et al. [16] describe the networking properties of Bitcoin and propose a multi-step attack for correlating Bitcoin clients' transactions with their IP addresses. The attack proceeds as follows. Firstly, the attacker prevents clients from using Tor by abusing the Bitcoin's anti-DoS mechanism: by sending invalid blocks or transactions through Tor it is possible to make Bitcoin servers temporarily ban all Tor exit nodes (see also [17]). Next, the attacker establishes multiple connections to each of the servers and tracks which of them advertise an IP address of the victim client. The intuition is that the client's *entry nodes* will be the ones to advertise its IP address to the attacker (this is not guaranteed; the paper suggests ways to reduce noise in the resulting data). After constructing a mapping of client IP addresses to sets of their entry nodes, the attacker listens to new transactions and correlates them with the victim client, if they are broadcast from that client's entry nodes.

Miller et al. [35] exploit some peculiarities in the update mechanism for a known address database (`addrMan`) in the Bitcoin reference implementation to infer the underlying graph structure. Each Bitcoin node maintains a database of IP addresses of peers it knows, along with corresponding timestamps intended to reflect the peer's "freshness". Unintuitively, at the time of writing (2015), Bitcoin nodes only update timestamps for nodes they maintain outgoing connections with (at each message received). For incoming connections, the peer preserves the first timestamp relayed along with the address. The authors implement a tool that takes advantage of such rules to make quite an accurate guess of the topology of the Bitcoin network. After an update of Bitcoin Core in March 2015, this technique is no longer feasible.

Neudecker et al. [36] propose a timing analysis attack to infer the network topology. Their approach is different from the previous work (and similar to ours) in that it does not use any side-channels, but only the timing of transaction propagation. The real-world validation in the Bitcoin network inferred network links at a substantial recall and precision. The authors showed that an inappropriately parameterized trickling mechanism can actually reduce the resistance to traffic analysis compared to naïve gossip (for the goal of learning the network topology).

Wang and Pustogarov [52] conduct a measurement study of Bitcoin to analyze the unreachable nodes (i.e., those behind NATs and firewalls) and report, among other findings, that a large share of Bitcoin transactions originate from only two mobile applications.

Fanti et al. [26] study the anonymity properties of trickling and diffusion. Despite the motivation to change the Bitcoin's propagation mechanism from trickling to diffusion, as the study shows, this provided only a marginal privacy improvement. The authors conclude that the key feature that enables deanonymization in both trickling and diffusion is an inherent symmetry: as messages spread through the network in a circular fashion, a global adversary can estimate where the center (i.e., the message source) is.

Dandelion [51] and its improvement Dandelion++ [25] are message propagation protocols for P2P networks designed to prevent deanonymization attacks. Its key idea is introducing asymmetry: a message is first sent along a random path, and only then broadcast gossip-style. Message propagation in Dandelion++[6] proceeds in two stages: the "stem" phase and the "fluff" stage. In the stem phase, a new message is broadcast along a random path in the anonymity graph: an approximately regular random graph based on the same set of nodes as the regular P2P network. In the fluff phase, the latest node to receive the message disperses it using the regular gossip-style broadcast. The authors show that the protocol achieves much stronger anonymity than Bitcoin's current propagation mechanism, though at the cost of a several second propagation delay and additional sensitivity to DoS attacks at stem phase.

Though the authors mention (Section 4.2) that some configurations of the protocol may be prone to transaction correlation attacks, our approach is not suitable against Dandelion++. The key feature that allows our well-connected listening node to gather useful information is that nodes choose neighbors to propagate messages at random, without distinguishing incoming and outgoing connections. This means that by saturating 50% of a node's connection slots we have a 50% chance to be the first to receive a new transaction from it. In Dandelion++, nodes choose neighbors for the stem phase propagation only from outgoing connections. There is no obvious way to force a remote peer to initiate a connection to us, therefore a malicious node with many outgoing connections will not have any advantage in the stem phase (it can only aggregate incoming information while acting as a regular relay, which may gain some but not much insight into possible transaction clusters).

Neudecker and Hartenstein [37] combine blockchain and network analysis to cluster Bitcoin addresses and associate them with IP addresses. They determine the originator of a transaction as the first originator, using two independent listening nodes and some heuristics to make the estimation more precise. The authors conclude that for the majority of

[6]We focus on the latest, improved version of the protocol.

users network-based deanonymization is not a concern, though a small percentage of users might be susceptible to attacks of this type.

V. Conclusion and future work

We study the state of anonymity of cryptocurrencies on the network level. We describe and implement a novel kind of transaction clustering based on the analysis of propagation times. We implement and test our tool on four popular cryptocurrencies: Bitcoin, Zcash, Dash, and Monero. Our results indicate that many cryptocurrencies, including privacy-focused ones, do not sufficiently defend against our attack: a low budget adversary can link transactions initially broadcast from the same node with a high degree of accuracy. We argue that cryptocurrencies must defend against network analysis to provide stronger privacy guarantees.

A. The applicability of the external quality metric

The adjusted anonymity degree, which we used as an external quality metric, has limitations. In particular, we didn't account for transactions from clusters which did not also contain at least one of our own transactions. The rationale behind this is the lack of the ground truth for two "foreign" transactions: we do not know whether they should be included in the same cluster. Consequently, our quality metric may poorly reflect the reality on large networks (such as the Bitcoin mainnet), where our transactions make up only a small part of the full network throughput. One direction of future research may be deriving an anonymity metric which works better under these circumstances.

B. Direct comparison of relay randomization techniques

As described in Section II, cryptocurrencies use different relay randomization techniques aimed at improving privacy: trickling, diffusion, or no randomization. A natural question would be to measure the relative effectiveness of these methods. Unfortunately, we cannot use a direct comparison between cryptocurrencies that use diffusion and trickling to make a conclusion about relative effectiveness of these methods, as the real-world networks also differ in many other parameters (such as the number of nodes and transaction rate) that also influence the attack results. A possible direction for future research may be to quantify the effects of trickling and diffusion on privacy properties of a Bitcoin-like cryptocurrency with respect to our attack technique, holding all other parameters equal.

References

[1] "Dash 0.12.0 release notes," 2015, https://github.com/dashpay/dash/blob/master/doc/release-notes/dash/release-notes-0.12.0.md.

[2] "Bitcoin protocol documentation," 2018, https://en.bitcoin.it/wiki/Protocol_documentation.

[3] "Bitcoin wallet," 2018, https://bitcoin.org/en/wallets/mobile/android/bitcoinwallet/.

[4] "Dash is digital cash," 2018, https://www.dash.org/.

[5] "Dash. protocol documentation - 0.12.1," 2018, https://github.com/dashpay/dash/blob/master/dash-docs/protocol-documentation.md.

[6] "Internet money," 2018, https://z.cash/.

[7] "Kovri," 2018, https://getkovri.org/.

[8] "Monero active nodes distribution," 2018, https://monerohash.com/nodes-distribution.html.

[9] "Monero. private digital currency," 2018, https://getmonero.org/.

[10] "Understanding masternodes," 2018, https://docs.dash.org/en/latest/masternodes/understanding.html.

[11] "Tor," 2019, https://www.torproject.org/.

[12] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo, "A comparison of extrinsic clustering evaluation metrics based on formal constraints," *Inf. Retr.*, vol. 12, no. 4, pp. 461–486, 2009.

[13] E. Androulaki, G. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in Bitcoin," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 7859. Springer, 2013, pp. 34–51, https://eprint.iacr.org/2012/596.

[14] E. Ben-Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from Bitcoin," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2014, pp. 459–474, .

[15] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von Neumann architecture," in *USENIX Security Symposium*. USENIX Association, 2014, pp. 781–796, http://zerocash-project.org/paper.

[16] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in Bitcoin P2P network," in *ACM Conference on Computer and Communications Security*. ACM, 2014, pp. 15–29, https://arxiv.org/abs/1405.7418.

[17] A. Biryukov and I. Pustogarov, "Bitcoin over Tor isn't a good idea," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2015, pp. 122–134, https://arxiv.org/abs/1410.6079.

[18] J. Bonneau, A. Narayanan, A. Miller, J. Clark, J. A. Kroll, and E. W. Felten, "Mixcoin: Anonymity for Bitcoin with accountable mixes," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 8437. Springer, 2014, pp. 486–504.

[19] J. Cameron, "What privacy issues did Monero have and still has?" 2016, https://monero.stackexchange.com/q/1495/4089.

[20] J. Connell, "How much does it cost to run a full Bitcoin node?" 2017, https://news.bitcoin.com/cost-full-bitcoin-node/.

[21] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *KDD*. ACM, 2001, pp. 269–274.

[22] C. Díaz, S. Seys, J. Claessens, and B. Preneel, "Towards measuring anonymity," in *Privacy Enhancing Technologies*, ser. Lecture Notes in Computer Science, vol. 2482. Springer, 2002, pp. 54–68.

[23] dpzz, "What's the difference between "balance" and "unlocked balance"?" 2017, https://monero.stackexchange.com/q/3262/4089.

[24] expez, "In what ways can a wallet connected to a malicious remote node be abused?" 2016, https://monero.stackexchange.com/q/2962/4089.

[25] G. C. Fanti, S. B. Venkatakrishnan, S. Bakshi, B. Denby, S. Bhargava, A. Miller, and P. Viswanath, "Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees," in *SIGMETRICS (Abstracts)*. ACM, 2018, pp. 5–7, https://arxiv.org/abs/1805.11060.

[26] G. C. Fanti and P. Viswanath, "Anonymity properties of the Bitcoin P2P network," *CoRR*, vol. abs/1703.08761, 2017, https://arxiv.org/abs/1703.08761.

[27] D. Hopwood, S. Bowe, T. Hornby, and N. Wilcox, "Zcash protocol specification," 2018, https://github.com/zcash/zips/blob/master/protocol/protocol.pdf.

[28] H. A. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "Blocksci: Design and applications of a blockchain analysis platform," *CoRR*, vol. abs/1709.02489, 2017, https://arxiv.org/abs/1709.02489.

[29] G. Kappos, H. Yousaf, M. Maller, and S. Meiklejohn, "An empirical analysis of anonymity in Zcash," *CoRR*, vol. abs/1805.03180, 2018, https://arxiv.org/abs/1805.03180.

[30] P. Koshy, D. Koshy, and P. D. McDaniel, "An analysis of anonymity in Bitcoin using P2P network traffic," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 8437. Springer, 2014, pp. 469–485, https://pdfs.semanticscholar.org/c277/62257f068fdbb2ad34e8f787d8af13fac7d1.pdf.

[31] manontheinside, "Does Monero protect against timing analysis?" 2016, https://monero.stackexchange.com/q/2765/4089.

[32] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," 2013, https://bitcointalk.org/index.php?topic=279249.

[33] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," *Commun. ACM*, vol. 59, no. 4, pp. 86–93, 2016.

[34] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2013, pp. 397–411, .
[35] A. Miller, J. Litton, A. Pachulski, N. Gupta, D. Levin, N. Spring, and B. Bhattacharjee, "Discovering Bitcoin's public topology and influential nodes," *et al.*, 2015, https://www.cs.umd.edu/projects/coinscope/coinscope.pdf.
[36] T. Neudecker, P. Andelfinger, and H. Hartenstein, "Timing analysis for inferring the topology of the Bitcoin peer-to-peer network," in *UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld*. IEEE Computer Society, 2016, pp. 358–367, https://dsn.tm.kit.edu/publications/files/323/bitcoin_timing_analysis_dsn.pdf.
[37] T. Neudecker and H. Hartenstein, "Could network information facilitate address clustering in Bitcoin?" in *Financial Cryptography Workshops*, ser. Lecture Notes in Computer Science, vol. 10323. Springer, 2017, pp. 155–169.
[38] M. Ober, S. Katzenbeisser, and K. Hamacher, "Structure and anonymity of the Bitcoin transaction graph," *Future Internet*, vol. 5, no. 2, pp. 237–250, 2013.
[39] A. M. Piotrowska, J. Hayes, T. Elahi, S. Meiser, and G. Danezis, "The Loopix anonymity system," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, 2017, pp. 1199–1216. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/piotrowska
[40] I. Pustogarov, "Bitcoin network probing tool," 2017, https://github.com/ivanpustogarov/bcclient.
[41] J. Quesnelle, "On the linkability of Zcash transactions," *CoRR*, vol. abs/1712.01210, 2017, https://arxiv.org/abs/1712.01210.
[42] F. Reid and M. Harrigan, "An analysis of anonymity in the Bitcoin system," in *SocialCom/PASSAT*. IEEE, 2011, pp. 1318–1326, https://arxiv.org/abs/1107.4524.
[43] D. Ron and A. Shamir, "Quantitative analysis of the full Bitcoin transaction graph," in *Financial Cryptography*, ser. Lecture Notes in Computer Science, vol. 7859. Springer, 2013, pp. 6–24.
[44] T. Ruffing, P. Moreno-Sanchez, and A. Kate, "Coinshuffle: Practical decentralized coin mixing for Bitcoin," in *ESORICS (2)*, ser. Lecture Notes in Computer Science, vol. 8713. Springer, 2014, pp. 345–364.
[45] rukoom, "Is Monero in I2P secure now, and how do I do it?" 2017, https://monero.stackexchange.com/q/6264/4089.
[46] scikit learn, "Biclustering," 2018, http://scikit-learn.org/stable/modules/biclustering.html.
[47] R. Spagni, "Monero 0.13.0 "beryllium bullet" release," 2018, https://www.getmonero.org/2018/10/11/monero-0.13.0-released.html.
[48] user36303, "Does each node have a maximum number of peers?" 2016, https://monero.stackexchange.com/a/1127/4089.
[49] L. Valenta and B. Rowan, "Blindcoin: Blinded, accountable mixes for bitcoin," in *Financial Cryptography Workshops*, ser. Lecture Notes in Computer Science, vol. 8976. Springer, 2015, pp. 112–126.
[50] N. van Saberhagen, "Cryptonote v 2.0," 2013, https://cryptonote.org/whitepaper.pdf.
[51] S. B. Venkatakrishnan, G. C. Fanti, and P. Viswanath, "Dandelion: Redesigning the Bitcoin network for anonymity," *POMACS*, vol. 1, no. 1, pp. 22:1–22:34, 2017, https://arxiv.org/abs/1701.04439.
[52] L. Wang and I. Pustogarov, "Towards better understanding of Bitcoin unreachable peers," *CoRR*, vol. abs/1709.06837, 2017, https://arxiv.org/abs/1709.06837.
[53] P. Wuille, "Replace trickle nodes with per-node/message poisson delays," 2015, https://github.com/bitcoin/bitcoin/commit/10b88be.
[54] Zcash, "Sapling," 2018, https://z.cash/upgrade/sapling.
[55] R. Zeyde, "Bitcoin full node on AWS free tier," 2018, https://gist.github.com/romanz/17ff716f13a34df49ff4.