Transaction Clustering Using Network Traffic Analysis for Bitcoin and Derived Blockchains

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

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

***Abstract*—Bitcoin is a decentralized digital currency introduced in 2008 and launched in 2009. Bitcoin provides a way to transact without any trusted intermediary, but its privacy guarantees are questionable, and multiple deanonymization attacks have been proposed. Cryptocurrency privacy research has been mostly focused on blockchain analysis, i.e., extracting information from the transaction graph. We focus on another vector for privacy attacks: network analysis.**

We describe the message propagation mechanics in Bitcoin and propose a novel technique for transaction clustering based on network traffic analysis. We show that timings of transaction messages leak information about their origin, which can be exploited by a well connected adversarial node. We implement and evaluate our method in the Bitcoin testnet with a high level of accuracy, deanonymizing our own transactions issued from a desktop wallet (Bitcoin Core) and from a mobile (Mycelium) wallet. Compared to existing approaches, we leverage the propagation information from multiple peers, which allows us to overcome an anti-deanonymization technique ("diffusion") used in Bitcoin.

***Index Terms*—Bitcoin, blockchain, cryptocurrency, privacy**

I. Introduction

Bitcoin is a decentralized digital currency launched in 2009. Bitcoin addresses are not linked to any real-world identity at the protocol level. Bitcoin does not guarantee absolute privacy though. Transactions are broadcast in plaintext through a peer-to-peer network. After being confirmed by miners they are stored in a massively replicated database (the blockchain). Deanonymization techniques based on blockchain analysis [13][18][21] and network analysis [2][8] have been developed. A number of alternative cryptocurrencies (most notably, Dash, Monero, and Zcash) address the privacy issue with more sophisticated cryptographic methods, which hide details of transactions (sender, receiver, amount) while still allowing anyone to verify their validity.

II. Transaction propagation in cryptocurrencies

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 required TCP port[1] 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 with a 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 Bitcoin and the major privacy-focused cryptocurrencies, we distinguish three propagation mechanisms:

- Naive 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).

III. Our Transaction Clustering Method

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.

[1] 8333 for the Bitcoin mainnet, 18333 for the Bitcoin testnet.

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` [19] to maintain parallel connections to peers and log incoming messages: 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 subsequent propagations).

3) Weight functions and clustering: Let tx be a transaction. Let $p^{tx} = [p_1^{tx}, p_2^{tx}, \dots p_N^{tx}]$ be the vector of the first N IP addresses which relayed tx to us. Let $t^{tx} = [t_1^{tx}, t_2^{tx}, \dots t_N^{tx}]$ 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:

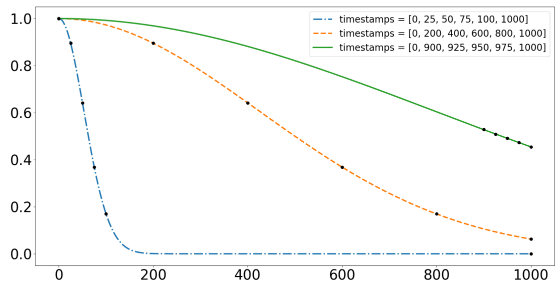

Fig. 1: Weight functions for three timestamp vectors

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

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. 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 [5] implemented in the Python `sklearn.cluster.bicluster` module [23]. 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 [1] (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 our transactions in the same cluster)[2];
- SD: same cluster, different category (our and foreign transactions in the same cluster);
- DS: different cluster, same category (two 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:

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

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. [6]. 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:

$$H_M = log_2(N)$$

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

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.

IV. Experiments

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. We discard transactions which are relayed to us by less than 10 peers or which were last relayed to us earlier than 30 seconds after the logging started (this likely means their propagation started before the experiment, and the relevant information is not recorded). For each transaction i, we assign an a priori probability of having originated from $S_{control}$: $p_i = n/N$. For wallets with P2P broadcast, the outline of our experiment is as follows.

1) establish parallel connections to a set of live peers, log the timestamps of incoming messages;
2) launch two nodes S_{learn} and $S_{control}$;
3) issue two series of transactions (the learning and the control sets) from S_{learn} and $S_{control}$ respectively;
4) calculate the transaction correlation matrix w.r.t. the weights of propagation times;
5) run the clustering algorithm with multiple sets of parameters and choose the best clustering by Rand score on the "learning" set;
6) in the best clustering, assign the cluster weights proportionally to the distribution of k known transactions from $S_{control}$ (transactions from S_{learn} get a zero probability weight);
7) calculate the final adjusted anonymity score w.r.t. the probability distribution among clusters;

8) visualize the results as a heatmap.

The results for the Bitcoin testnet are presented in Figure 2. In that experiment (November 2018), we tried to connect to all available peers. We established around 64 connections on average to 1129 out of 1142 peers from a fresh snapshot. We logged the traffic for 20 minutes and captured inventory messages for one block, 402 addresses, and 139 transactions. Our learning set and control set consisted of 30 transactions each, issued from a standard Bitcoin Core client.

The correlation matrix shows a block-diagonal structure, as expected. Of the 30 transactions from the control set, 25 ended up in one cluster, which also contained 7 unrelated transactions (78% precision and 83% recall). Assume an adversary who knows that some of the transactions originate from the victim node. From the clustering picture, the adversary can deduct, with a high level of confidence, which other transactions, issued around that time, originate from the same node.

To validate our method for the case when the victim issued transactions from a mobile device, we conducted an experiment using a testnet version of a popular Mycelium wallet for Android as the source of the transactions in the control set. We established around 64 connections on average to 1126 out of 1133 peers from a fresh snapshot. We logged the traffic for 20 minutes and captured inventory messages for 3 blocks, 320 addresses, and 115 transactions.

The results for Mycelium are presented in Figure 3. The first cluster contains all 28 control set transactions[3], and 13 other transactions, which yields 100% precision and 68% recall. Note that Mycelium does not connect directly to the Bitcoin P2P network. Transactions are sent to a server via TLS, and then broadcast by the node(s) maintained by the wallet developers. This means that in a real-world scenario a cluster corresponding to a wallet with a centralized broadcast (the majority of mobile wallets) corresponds to transactions issued by all users of this wallet in the relevant timeframe and broadcast from a single node.

V. Discussion

We now discuss the possible attack scenarios and countermeasures.

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

A popular mitigation technique against networking-based deanonymization attacks is to use anonymity overlay networks such as Tor. 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 another anonymity network before being publicly broadcast. 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

[3]One additional transaction from the control set was not captured by our listening node.

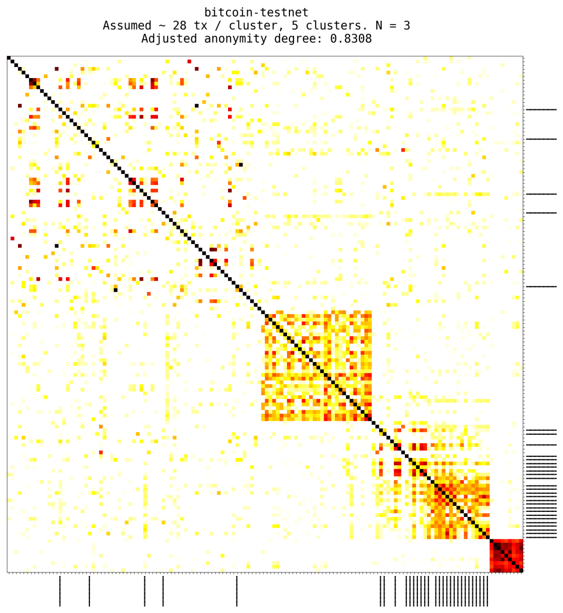

Fig. 2: Bitcoin Core (testnet). Black dashes correspond to our control set transactions.

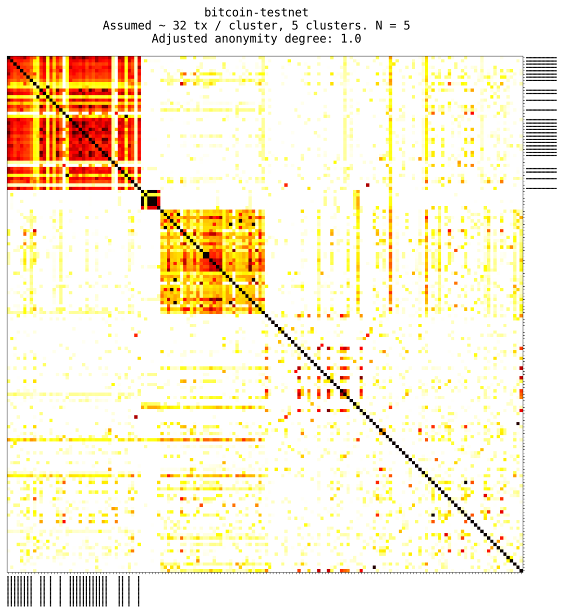

Fig. 3: Mycelium (testnet)

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 [3] (the situation is similar to the case of a light wallet described below).

Below we distinguish three cases depending on the type of the user's node and wallet and suggest countermeasures against our attack.

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 or from the same wallet session.

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, such 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 other full node using simple payment verification (SPV). From the networking perspective, most light wallets, especially mobile, 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[4]. 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);
- 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.

Cryptocurrency developers should introduce privacy-enhancing measures on 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 [25][7].

VI. Related work

Early research on cryptocurrency privacy mostly covered Bitcoin public blockchain graph analysis.A popular mitigation technique is mixing: users combine inputs in a joint transaction, making it harder for an adversary to track the flow of coins. The major drawback of initial mixing proposals is that users must agree to co-sign the transaction out-of-band. Various mixing schemes have been proposed [4][24][12][22]. Gervais et al. [9] analyze the privacy implications of Bloom filters in SPV wallets. Hussain et al. [10] propose a smart card based security extension to Bitcoin wallets. Montanez [15] analyze Bitcoin wallets for iOS and Android from the forensics perspective.

Approaches to deanonymization using network analysis developed from a simple "first relayer" heuristic [11] to more sophisticated techniques: fingerprinting by entry set [2] (while abusing the Bitcoin DoS-protection mechanism to prevent the victim from connecting over Tor [3]), exploiting peculiarities in the update mechanism for known address database [14], discovering the network topology from timing analysis [16]. These techniques are being applied to privacy-focused cryptocurrencies as well [20]. Neudecker and Hartenstein [17] combine blockchain and network analysis to cluster Bitcoin addresses and associate them with IP addresses. Dandelion [25] and its improvement Dandelion++ [7] are message propagation protocols for P2P networks designed to prevent deanonymization attacks by introducing asymmetry in message propagation.

VII. Conclusion

We proposed and tested a transaction clustering technique based on propagation timing. We showed that a global passive adversary can cluster transactions issued from one device within a short time frame with relatively high accuracy.

Most alternative cryptocurrencies use similar networking mechanics, many of them originating from forks of the Bitcoin Core codebase. Out of the major alternative cryptocurrencies, Litecoin and Zcash directly inherit networking mechanics from Bitcoin. Dash, being initially based on Bitcoin Core code, utilizes many additional message types and uses a two-tier network with masternodes transmitting messages alongside usual nodes. Monero, a privacy-focused cryptocurrency implemented independently of Bitcoin, uses a similar peer-to-peer gossip protocol without trickling or diffusion. This

[4]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 user's transactions and link them to their IP address. Using Tor is not applicable in this case, as the wallet servers will still be able to associate the user's transactions by other means (e.g., by making the wallet send a cookie along with transactions).

suggests that our technique in addition to Bitcoin is applicable for alternative cryptocurrencies as well.

We suggest implementing more strong anonymization techniques on the networking level in Bitcoin and other cryptocurrencies. Cryptocurrency users should be aware of the issue and not broadcast sensitive transactions within a short timeframe using the same set of entry nodes.

References

[1] Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Inf. Retr.*, 12(4):461–486, 2009.

[2] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in Bitcoin P2P network. In *ACM Conference on Computer and Communications Security*, pages 15–29. ACM, 2014. https://arxiv.org/abs/1405.7418.

[3] Alex Biryukov and Ivan Pustogarov. Bitcoin over Tor isn't a good idea. In *IEEE Symposium on Security and Privacy*, pages 122–134. IEEE Computer Society, 2015. https://arxiv.org/abs/1410.6079.

[4] Joseph Bonneau, Arvind Narayanan, Andrew Miller, Jeremy Clark, Joshua A. Kroll, and Edward W. Felten. Mixcoin: Anonymity for Bitcoin with accountable mixes. In *Financial Cryptography*, volume 8437 of *Lecture Notes in Computer Science*, pages 486–504. Springer, 2014.

[5] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *KDD*, pages 269–274. ACM, 2001.

[6] Claudia Díaz, Stefaan Seys, Joris Claessens, and Bart Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 2002.

[7] Giulia C. Fanti, Shaileshh Bojja Venkatakrishnan, Surya Bakshi, Bradley Denby, Shruti Bhargava, Andrew Miller, and Pramod Viswanath. Dandelion++: Lightweight cryptocurrency networking with formal anonymity guarantees. In *SIGMETRICS (Abstracts)*, pages 5–7. ACM, 2018. https://arxiv.org/abs/1805.11060.

[8] Giulia C. Fanti and Pramod Viswanath. Anonymity properties of the Bitcoin P2P network. *CoRR*, abs/1703.08761, 2017. https://arxiv.org/abs/1703.08761.

[9] Arthur Gervais, Srdjan Capkun, Ghassan O Karame, and Damian Gruber. On the privacy provisions of Bloom filters in lightweight Bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 326–335. ACM, 2014.

[10] Majid Amjad Hussain, Sadia Khalil, and Shahzad Saleem. A smart card based security extension for the bitcoin wallets. *NUST Journal of Engineering Sciences*, 9(2):60–67, 2016.

[11] Philip Koshy, Diana Koshy, and Patrick D. McDaniel. An analysis of anonymity in Bitcoin using P2P network traffic. In *Financial Cryptography*, volume 8437 of *Lecture Notes in Computer Science*, pages 469–485. Springer, 2014.

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

[13] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. *Commun. ACM*, 59(4):86–93, 2016.

[14] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering Bitcoin's public topology and influential nodes. *et al.*, 2015. https://www.cs.umd.edu/projects/coinscope/coinscope.pdf.

[15] Angelica Montanez. Investigation of cryptocurrency wallets on iOS and Android mobile devices for potential forensic artifacts. *Dept. Forensic Sci., Marshall Univ., Huntington, WV, USA, Tech. Rep*, 2014.

[16] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the Bitcoin peer-to-peer network. In *UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld*, pages 358–367. IEEE Computer Society, 2016.

[17] Till Neudecker and Hannes Hartenstein. Could network information facilitate address clustering in Bitcoin? In *Financial Cryptography Workshops*, volume 10323 of *Lecture Notes in Computer Science*, pages 155–169. Springer, 2017.

[18] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the Bitcoin transaction graph. *Future Internet*, 5(2):237–250, 2013.

[19] Ivan Pustogarov. Bitcoin network probing tool, 2017. https://github.com/ivanpustogarov/bcclient.

[20] Jeffrey Quesnelle. On the linkability of Zcash transactions. *CoRR*, abs/1712.01210, 2017. https://arxiv.org/abs/1712.01210.

[21] Dorit Ron and Adi Shamir. Quantitative analysis of the full Bitcoin transaction graph. In *Financial Cryptography*, volume 7859 of *Lecture Notes in Computer Science*, pages 6–24. Springer, 2013.

[22] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for Bitcoin. In *ESORICS (2)*, volume 8713 of *Lecture Notes in Computer Science*, pages 345–364. Springer, 2014.

[23] scikit learn. Biclustering, 2018. http://scikit-learn.org/stable/modules/biclustering.html.

[24] Luke Valenta and Brendan Rowan. Blindcoin: Blinded, accountable mixes for Bitcoin. In *Financial Cryptography Workshops*, volume 8976 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2015.

[25] Shaileshh Bojja Venkatakrishnan, Giulia C. Fanti, and Pramod Viswanath. Dandelion: Redesigning the Bitcoin network for anonymity. *POMACS*, 2017. https://arxiv.org/abs/1701.04439.