Security and Privacy of Mobile Wallet Users in Bitcoin, Dash, Monero, and Zcash

Alex Biryukov[a], Sergei Tikhomirov[a]

[a] *University of Luxembourg*

Abstract

Mobile devices play an increasingly important role in the cryptocurrency ecosystem, yet their privacy guarantees remain unstudied. To verify transactions, they either trust a server or use simple payment verification.

First, we review the security and privacy of popular Android wallets for Bitcoin and the three major privacy-focused cryptocurrencies (Dash, Monero, Zcash). Then, we investigate the network-level properties of cryptocurrencies and propose a method of transaction clustering based on timing analysis. We implement and test our method on selected wallets and show that a moderately resourceful attacker can correlate transactions issued from one device with relatively high accuracy.

Keywords: blockchain, cryptocurrency, privacy, Android

1. Introduction

Bitcoin is a decentralized digital currency launched in 2009. Multiple alternative cryptocurrencies have been proposed since. A cryptocurrency is based on P2P network of *nodes*. *Full* nodes download and validate the whole blockchain. *Light*, or *thin*, nodes query information from other nodes

Email addresses: `alex.biryukov@uni.lu` (Alex Biryukov), `sergey.s.tikhomirov@gmail.com` (Sergei Tikhomirov)

to save storage space and bandwidth. Light nodes present trade-offs in terms of security and privacy. First, in case of a successful eclipse attack, an adversary can deny service or provide false information. Therefore, it is important for light nodes to ensure that at least one connected full node is honest. This is achieved by either connecting to a random set of nodes (bootstrapping), or to a trusted node. Second, an "honest but curious" node or a global passive adversary can deduct a user's Bitcoin addresses based on Bloom filters [14].

Cryptocurrency wallets on mobile devices have limited resources, therefore they either fully trust a centralized server for blockchain data, or use the simple payment verification protocol (SPV). An *SPV node* connects to full nodes and downloads only block headers, assuming the header with the most proof-of-work is the valid one. Full nodes provide light nodes with transactions information along with a Merkle proof of their membership in the heaviest chain. To provide stronger privacy, SPV nodes issue queries using Bloom filters with a specified false positive rate (and zero false negatives), and discard unnecessary data client-side.

We are using the term "privacy" to denote the situation where details of a person's cryptocurrency transactions are hidden from third parties. Related concepts are anonymity and confidentiality. In Section 2.1, we consider wallet developers as adversaries trying to collect data about their users' transactions. Through studying the features of prominent mobile wallet, we try to estimate the amount of personal information their developers can collect. We evaluate the security and privacy of popular Android wallets for various cryptocurrencies. We formulate four criteria for a minimally privacy-preserving wallet and apply them to a selection of wallets. Then, in Section 3, we study the source code of selected wallets, both manually and using static analysis tools, to detect potential to privacy violations. Then, in Section 4, we consider an adversary who is listening to network traffic and seeks to correlate cryptocurrency addresses with IP addresses of nodes or other identifying information. We present a deanonymization algorithm that clusters transactions originating from the same node. We evaluate its efficiency on selected wallets and provide a list of recommendations to users and developers. Section 7 provides a review of related work, and Section 8 concludes.

2. Security and privacy of mobile wallets

We distinguish two types of mobile wallets from a networking perspective. Wallets with *centralized broadcast* (referred to as *centralized wallets*) send transactions to a server maintained by the wallet developers, which broadcasts them to the P2P network. Wallets with *P2P broadcast* (*P2P wallets*) connect to peers directly.

We compile a list of wallets recommended on the official websites of the considered cryptocurrencies: Bitcoin, Dash, Monero, and Zcash. We add to the list the most popular wallets according to Google Play[1]). Additionally, we also consider Samourai – a Bitcoin wallet specifically advertised as privacy-focused. Note that Samourai connects to a selected trusted node via RPC, not P2P, and requires full control over it [23] (marked with "+*"

[1]Popular wallets not mentioned on any official websites are Bitcoin wallet by Bitcoin.com (referred to as Bitcoin.com), Bitpay, and Copay. Copay is an open-source wallet developed by Bitpay. The same company also offers the Bitpay wallet, which is based on the source code of Copay with some additional services. Due to the similarity of these wallets, we only considered Copay.

in Table 1).

2.1. Initial privacy criteria

Paying with cryptocurrency involves signing a transaction and sending it to the P2P network for miners to confirm. To preserve privacy, a user should not be required to provide any information to the wallet provider. To ensure the lack of backdoors or other unintended functionality, a wallet should also be open-sourced, which is a common requirement for security-related software. Consequently, we argue that the following criteria are minimally necessary for a mobile wallet[2]:

1. No registration required. Otherwise, the user's transactions can be deanonymized by the wallet provider. We check this criterion by installing a wallet and attempting to generate a receiving address without providing any personal information.
2. Open-sourced code. A malicious closed-source application can track users or even steal their funds. We consider this criterion met if there is a publicly available code repository with a fully fledged Android application linked from the wallet's official website.
3. Private keys generated locally. A hosted wallet (which stores keys on a server) requires full trust.
4. P2P networking. The wallet must request blockchain data and broadcast transactions via the P2P network directly, not via a trusted server, which can deanonymize and censor transactions.

We check the criteria 3 and 4 based on the official documentation and the source code, if available. The following wallets pass our initial test (Bitcoin

[2]For wallets which require registration, we may not have checked all other criteria.

unless specified): Bitcoin wallet, Bither, BRD, Dash wallet (Dash), Electrum[3], Monerujo (Monero), Simple Bitcoin (see Table 1). Zcash has no Android wallets that satisfy our privacy criteria. No multi-currency wallets satisfy our criteria.

3. Analysis of selected wallets

We compare the wallets which passed our initial criteria manually and using static analysis tools.

3.1. Manual inspection

Independent installation. The most prevalent way of installing Android apps is the Google Play store. A privacy-conscious user may want to avoid linking their cryptocurrency activity with their Google profile. Alternative ways to install Android apps are F-Droid (an independent app store for free and open source apps) or manual installation from an APK file.

Permissions. Android *permissions* restrict access to certain user information and device functionality. Each application declares the necessary permissions in its *manifest file.* The user grants permissions at installation time or at runtime (starting from Android 6.0). Permissions that give access to critical functionality or personal information are refereed to as *dangerous.*

[3]Electrum is originally a desktop application; the official Github repository gives instructions on how to generate an APK file using Kivy GUI. Electrum wallet relies on an independent network of nodes (Electrum servers) to receive blockchain data and broadcast transactions. Though Electrum servers are not genuine cryptocurrency P2P nodes, we consider Electrum satisfy the P2P criteria, as a user can technically choose which Electrum servers to connect to, including their own.

[5] Airbitz requires the highest number of permissions (15). Two apps require access to both coarse and fine location (Airbitz, BRD), and sending SMS (BRD, Samourai). Electrum requests the lowest number of permissions: 3 dangerous and 1 other. All wallets require at least one dangerous permission: camera access is needed to scan cryptocurrency addresses presented as QR codes.

Privacy policies. All Android apps that handle "personal or sensitive user data" must have a privacy policy (PP). We compare the PPs of the selected wallets (see Table 2). Monerujo PP notes that the app uses the exchange rate information provided by the public API of `kraken.com`, and an exchange service `xmr.to`, which are subjects to their own privacy policies (marked with (+) in Table 2). Mycelium transmits a report to the developers' server in case of a crash; the developers claim they "took care that it does not contain unnecessary privacy relevant information". The Samourai wallet collects users' IP addresses "with replaced last byte", which can hardly be considered anonymization, as one may still infer the approximate location from the first three bytes of the IP address (marked with "+*" in Table 2). Airbitz broadcasts transactions through Electrum servers; a user may choose one or more trusted servers.

Connecting to the network. All wallets except Monerujo use hard-coded DNS seeds to bootstrap. BRD adds its own DNS seed[4]. Simple Bitcoin adds one random node from a hard-coded list[5] to a list of peers obtained via bootstrapping. Electrum connects to two random servers from a hard-

[4] `seed.breadwallet.com.`; does not resolve at the time of writing.

[5] `5.9.104.252`, `213.133.103.56`, `213.133.99.89`; all unreachable at the time of writing.

coded list of 52 servers; it requests the transaction history from a single server and checks it against block headers sent by other servers. Monerujo lets the user either choose from 3 hard-coded URLs which resolve to a list of publicly available nodes[6] or provide credentials for connecting to a custom node. Most wallets with P2P networking have a *network monitor* which displays the IP addresses of connected nodes.

3.2. Static analysis

3.2.1. FlowDroid

We scanned the wallets with FlowDroid [11] – an open source static analysis tool[7]. It uses data flow analysis to detect execution paths which transfer data from sources (functions which return sensitive data) into sinks (functions which send data elsewhere).

3.2.2. SmartDec Scanner

We scan the wallets using a proprietary static analysis tool SmartDec Scanner [3]. We manually inspected the results and summarize prevalent privacy-related issues detected by the tool, roughly in decreasing order of potential threat. Note that these issues do not directly lead to an exploit.

Leak to external storage (Electrum, Simple Bitcoin, Jaxx, Copay, Airbitz). Android provides internal and external storage[8]. An app can only access its

[6]`node.moneroworld.com:18089`, `node.xmrbackb.one`, `node.xmr.be`

[7]The analyses of Bitcoin wallet, Bither, and Monerujo were stopped after a 2 hour timeout. Samourai was not scanned due to a technical limitation: the app is labeled as unreleased, which prevented us from downloading the APK

[8]Historically, external storage was assumed to be on a removable memory card, now this may not be the case.

own directory in the itab:privacy-policiesnternal storage; external storage is available for other apps. Sensitive data should only be kept in the internal storage. Android automatically backs up data and settings of apps which did not opt out by setting `android:allowBackup=\false"` in the Manifest file. Automatic backups should be disabled for privacy-critical apps.

XSS attacks via Javascript in WebView (BRD, Bitcoin.com, Coinomi, Jaxx, Copay, Airbitz). One method of developing dynamic user interfaces on Android is using JavaScript inside a WebView (an Android component for displaying web pages). By default, execution of JavaScript code in WebView is disabled, but this setting can be overridden (`setJavaScriptEnabled(true)`). Executing malicious[9] JavaScript code may lead to cross-site scripting (XSS) and other attacks. We detect two instances of this issue (in `FragmentSupport` and `WebViewActivity` classes) in BRD. In both cases, the warning from Android Lint – a static analyzer built into the standard Android development environment – is suppressed.

Insecure connection (Bitcoin Wallet, Bither, Dash Wallet, Simple Bitcoin). Parameters of a TLS connection in Java are specified in the `X509TrustManager` class. Its methods can be overridden to accept all certificates (not only those authenticated by a chain of signatures up to a trusted root CA), which may lead to a man-in-the-middle attack. Connections between Electrum servers, unlike the Bitcoin protocol, are encrypted and authenticated with TLS. Bitcoin wallet and Dash wallet, though communicating with the respective networks via their P2P protocols, use Electrum servers for query-

[9]Even trusted code, e.g., from the app's resources, may contain unintended side effects or bugs, as well as implicitly leak information.

ing the balance when sweeping a paper wallet (see the `X509TrustManager`-inherited `RequestWalletBalanceTask` class). Bither defines HTTP URLs of its own API (`bither.net`) and a block explorer `blockchain.info` (class `BitherUrl`), which can lead to displaying incorrect balances or fake transactions in case of a man-in-the-middle attack. Simple Bitcoin uses five hard-coded URLs to query current fees; one of them[10] uses an unencrypted connection. This may lead to incorrect fee estimation (and thus a potential denial of service attack, as transactions with very low fees may never be confirmed), though this scenario requires four other APIs served over HTTPS to fail simultaneously.

Leak into logs (all wallets). Android applications can write to their own log. Applications can also read their own logs with the `READ_LOGS` permission (prior to Android 4.0, this also granted access to other apps' logs). It is possible to access logs on a rooted device, or using developer tools. All wallets log details about their operation, including error messages, which may include sensitive data (e.g., a trusted node's IP). This issue is present at least to some extent in all wallets, as all wallets use logging in exception handlers. One may find all occurrences of this issue by searching for the methods of the `Log` class, `print`, `println`, and exception handlers with `ex.printStackTrace()`. Further investigation is needed to determine the impact and probability of data leaks.

See Table 2 for the full results (BW – Bitcoin wallet, BH – Bither, BR – BRD, D – Dash wallet, El – Electrum, Mo – Monerujo, SB – Simple Bitcoin, Bc – Bitcoin wallet by Bitcoin.com, My – Mycelium, Co – Coinomi, Ja –

[10]`http://api.blockcypher.com/v1/btc/main`.

Table 1: Initial list of wallets (bold: four criteria met)

Wallet	Coin support				No reg	OS	Keys local	P2P
	BTC	DASH	XMR	ZEC				
Abra	+	+	+	+	-	-	+	?
Airbitz	+	-	-	-	-	+	+	?
ArcBit	+	-	-	-	+	+	+	-
Bitcoin.com	+	-	-	-	+	+	+	-
Bitcoin wallet	+	-	-	-	+	+	+	+
Bither	+	-	-	-	+	-	+	+
BTC.com	+	-	-	-	-	+	+	-
BRD	+	-	-	-	+	+	+	+
Coin.space	+	-	-	-	+	+	+	-
Coinomi	+	+	-	+	+	+	+	-
Copay	+	-	-	-	+	+	+	-
Dash wallet	-	+	-	-	+	+	+	+
Edge	+	+	+	-	-	-	+	-
Electrum	+	-	-	-	+	+	+	+
Ethos	+	+	+	+	-	-	?	?
GreenBits	+	-	-	-	+	+	+	-
Jaxx	+	+	-	+	+	-	+	-
Mobi.me	+	+	+	+	-	-	?	?
Monerujo	-	-	+	-	+	+	+	+
Mycelium	+	-	-	-	+	+	+	-
Samourai	+	-	-	-	+	+	+	+*
Simple Bitcoin	+	-	-	-	+	+	+	+
Zelcore	+	-	-	+	-	-	+	-

Table 2: Privacy policies of selected wallets

	BW	BH	BR	D	El	Mo	SB	Bc	My	Co	Ja	CP	AB	Sa
IP address	-	-	+	-	+	(+)	?	-	?	+	-	+	+	+*
browser version	-	-	+	-	-	(+)	?	-	?	+	-	?	+	+
pages visited	-	-	+	-	-	(+)	?	-	?	+	-	?	+	+
time of visit	-	-	+	-	+	(+)	?	-	?	+	-	?	+	+
unique device ID	-	-	+	-	-	(+)	?	-	?	-	-	?	?	-
other diagnostics	-	-	+	-	+	(+)	?	-	?	+	+	?	+	+
type of device	-	-	+	-	-	(+)	?	-	?	-	-	?	+	+
OS type	-	-	+	-	+	(+)	?	-	?	+	-	?	+	+
location	-	-	+	-	-	(+)	?	-	?	+	-	?	-	-
device name	-	-	-	-	+	-	?	-	?	-	-	?	?	-
app configuration	-	-	-	-	+	-	?	-	?	-	-	?	-	-
pages visited before	-	-	-	-	-	(+)	?	-	?	+	-	?	-	-
browser plug-ins	-	-	-	-	-	(+)	?	-	?	-	-	?	-	-
time zone	-	-	-	-	-	(+)	?	-	?	-	-	?	-	-
“clickstream”	-	-	-	-	-	(+)	?	-	?	-	-	?	-	-
cookies	-	-	+	-	-	-	?	-	?	-	+	?	+	+
analytics	-	-	+	-	+	-	?	-	-	-	-	?	+	+
Trusted node	+	-	-	+	+	+	+	-	-	-	-	-	+	+
Data leaks	0	4	3	1	0	2	1	0	6	4	0	0	4	?
F-Droid	+	-	-	-	-	+	+	-	-	-	-	-	-	-
APK download	+	+	-	-	+	+	+	+	+	+	-	-	-	-
Uses BitcoinJ	+	-	-	+	-	-	+	-	+	+	-	-	-	+
Net monitor	+	+	±	+	±	+	-							
Connections	4-6	6	3	4-6	2	1	10						+	

Jaxx, CP – Copay, AB – Airbitz, Sa – Samourai).

4. Transaction clustering

In this section, we discuss transaction clustering based on propagation timing. We describe the currently used transaction broadcast mechanisms in cryptocurrencies. We then present our method for transaction clustering based on the analysis of transaction propagation time. We test our method on selected mobile wallets with peer-to-peer as well as with centralized broadcast.

4.1. Transaction propagation in cryptocurrencies

Bitcoin Core attempts to establish 8 outgoing and allows 117 incoming connections by default. We refer to the set of connected nodes as the *entry set* (consisting of *entry nodes*). Let us assume node X has a new object (transaction or block) and wants to send it to an entry node Y. In Bitcoin Core and its forks, this is a three-step process:

1. X sends an inventory (`INV`) message to Y, specifying the object hash;
2. if Y is interested in receiving the object, it replies with a `GETDATA` message;
3. X sends the object in a `TX` or `BLOCK` message.

A simple P2P gossip in a cryptocurrency network may be harmful for the users' privacy: as messages propagate through the network in a "circular" fashion, a well-connected adversary may infer the "source of the rumor" by observing propagation delay at various nodes in the network. To prevent this threat, Bitcoin messages are broadcast to neighborinig nodes after a random delay (mechanism known as *diffusion*) [6]. Previously, another mechanism (*trickling*) was used, where messages were put in a queue and broadcast to a random subset of the neighbors. Both mechanisms provide only a marginal privacy improvement [13].

None of the P2P wallets we considered use diffusion or trickling. Most of them rely on the BitcoinJ library for networking. Unlike Bitcoin Core, BitcoinJ sends `TX` unconditionally[11]. The BitcoinJ developers acknowledge that the three-step `INV` – `GETDATA` – `TX` exchange in Bitcoin Core improves privacy, but argue that since BitcoinJ is used by light nodes with a priori weaker privacy guarantees, the three-step broadcast would only decrease efficiency. BitcoinJ and BRD broadcast a new transaction to a subset of entry nodes and wait for the announcement from other ones, which is taken as a sign of network acceptance.

4.2. Our approach

4.2.1. 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.

[11]Note that in this case a full node receiving a `TX` message from an SPV node can be sure that the transaction originated at that SPV node whereas receiving an `INV` announcement from a full node may also be a re-broadcast. This demonstrates the privacy enhancement of exclusively connecting to a trusted node for SPV.

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).

4.2.2. 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 propagations).

4.2.3. 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, the 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:

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

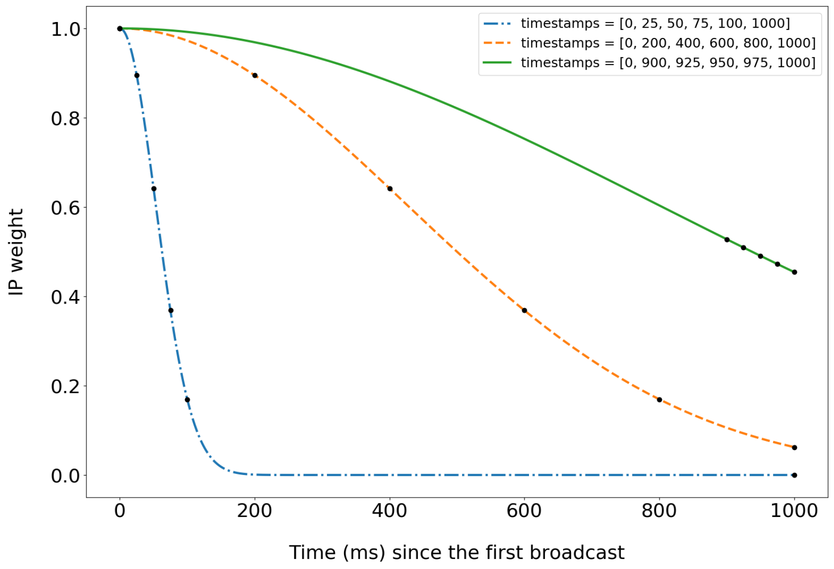


Figure 1: Weight functions for three timestamp vectors

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 [9] implemented in the Python `sklearn.cluster.bicluster` module [22]. 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.

4.3. Quality assessment

4.3.1. Measuring clustering quality

We use the Rand score as an external metric of clustering quality, as described in [4] (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);[12]

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

- 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:

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

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.

4.3.2. 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. [10]. 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.

5. 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 ($n > k$). 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);

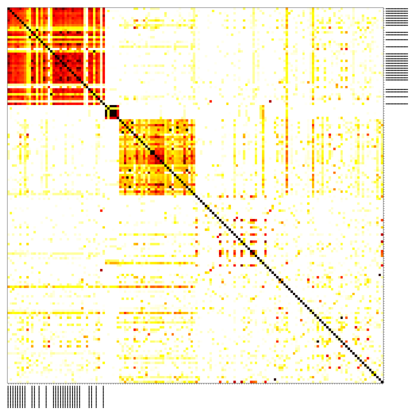
7. calculate the final adjusted anonymity score w.r.t. the probability distribution among clusters;
8. visualize the results as a heatmap.

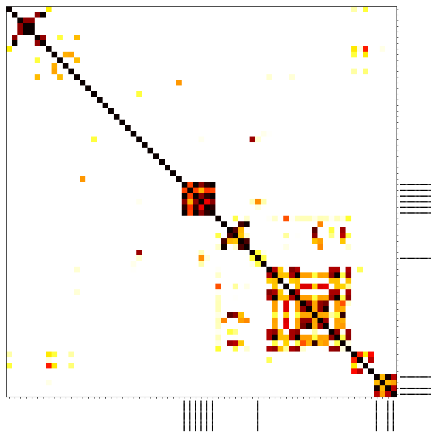
We focus on three networks for our experiments: Bitcoin testnet, Bitcoin mainnet, and Zcash mainnet. Bitcoin testnet is the most active testnet of an open cryptocurrency and is a perfect testing ground to perform fully-fledged experiments. We test the applicability of our technique on the two mainnets while not aiming to fully occupy the connection slots to avoid impacting the real networks. For the experiments we choose popular wallets with well-established brands (both Bitcoin-only and multi-coin), with centralized as well as P2P broadcast mechanisms.

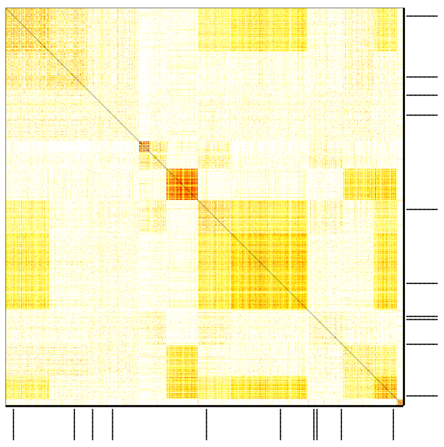
Our tool is not fully compatible with Dash and Monero. Dash, while based on a fork of Bitcoin Core, introduces many additional message types to manage the masternode network. Monero is implemented independently and uses another networking mechanism, although based on the same principles.

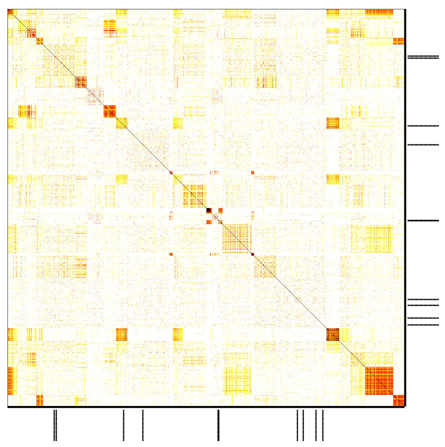
For centralized wallets, instead of issuing two sets of transactions, we only issue one set, using it as a “label” for a presumed wallet cluster. If our transactions form a visible cluster, we inspect the IP addresses of nodes which were among the first ones to broadcast them and assume those are the wallet nodes. This allows us to infer the IP addresses of nodes which a wallet uses for transaction broadcasts. We can then associate transactions in the network with popular wallets.

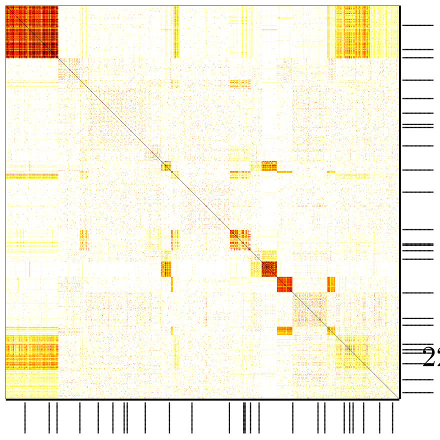
We performed experiments on Bitcoin testnet (Bitcoin wallet), Bitcoin mainnet (Bitcoin wallet, BRD, Coinomi, Mycelium), and Zcash (Coinomi). Bitcoin wallet and BRD use P2P broadcast; Coinomi and Mycelium use centralized broadcast. Experiments on testnet allow us to establish the


(a) Mycelium (testnet)


(b) Bitcoin wallet (testnet)


(c) Bitcoin wallet


(d) BRD


(e) Coinomi

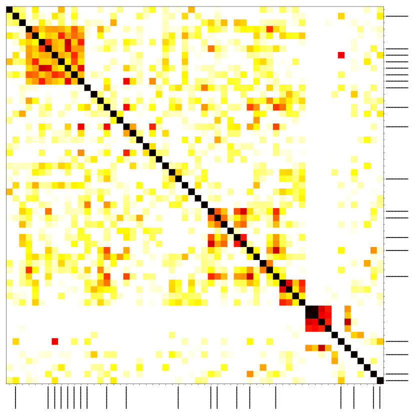

(f) Coinomi (Zcash)

Figure 2: Experimental results (Bitcoin unless specified)

upper bound on the effectiveness of our approach, as the testnet has a lower transaction rate and fewer nodes, which allows us to occupy all available connection slots. We considered the number of first propagations in the range from 3 to 7, $k = 5$.

The results are presented in Figure 2. Our (control) transactions are marked with black ticks. The color of each square represents the correlation coefficient between the weight vectors of two transactions (darker is higher). As expected, the matrices exhibit a block-diagonal structure, with clusters along the main diagonal corresponding to transaction sources. The clusters are clearly visible for the testnet version of Bitcoin wallet, we obtained a low (good for the attacker) anonymity degree of 0.5089. The adjusted anonymity degree (d_{adj}) for Bitcoin wallet (mainnet) is 0.8646, BRD (mainnet) – 0.8413, Coinomi – 0.9117. The experimental results show that while our technique shows the expected results on small networks, the picture for the Bitcoin mainnet is much less clear. This may be explained by a much larger total number of nodes and transactions, and to the fact that we only run the experiment on a subset of Bitcoin nodes to avoid disrupting the network.

5.1. Estimating the IP addresses of wallet's nodes

Apart from clustering transactions, an adversary might be interested in obtaining the IP address of the nodes which a centralized wallet uses for transaction broadcast. While the IP address of a centralized wallet's node may not necessarily be secret, the fact that a transaction from a particular Bitcoin address is propagated from such node may give an adversary a clue on which software the victim is using, which helps set up e.g. phishing or other social engineering attacks. We test this attack scenario in the experiments on Mycelium (Bitcoin testnet) and Coinomi (Zcash).

Mycelium transactions cluster well: they are quickly propagated from the same two IP addresses (delay in single milliseconds), rebroadcasts from other nodes follow after tens or even hundreds of milliseconds. According to IP geolocation services, these nodes are located in Germany (`2a01:4f9:2b:4ca::2`) and in Helsinki, Finland (`95.216.68.181`). According to a reverse DNS lookup service `robtex.com`, one of these IP addresses corresponds to a URL `electrumx-b.mycelium.com`. Both addresses belong to Hetzner (a cloud provider), have an uptime of 2 months (at the time of writing), and a latency of 25 ms, as per Bitnodes [2]. In a separate experiment, we discovered that each of them has more than 700 slots. We discovered that there are also Bitcoin mainnet nodes running at the same IP addresses.

In the Zcash experiment (Coinomi), though our transactions don't form a clear cluster, we observe that some of them (in the second cluster) were first received from the same IP (`5.79.123.194`), which, we assume, is at least one of the Coinomi's nodes.

6. Discussion

Experiments have shown that our approach works well in small networks (Bitcoin testnet, Zcash), but shows weaker results for Bitcoin mainnet. Bitcoin mainnet, being substantially larger than alternative networks, requires significant resources just to capture the traffic. Moreover, Bitcoin nodes have fewer free slots, and often limit the number of slots one IP can occupy. These factors combined with a higher overall transaction rate makes clustering in Bitcoin harder. Note also that we deliberately connected only to a small subset of the Bitcoin mainnet (up to 1000 out of approximately 10 thousand nodes) due to resource constraints. A more resourceful attacker may achieve better results. In addition, an adversary can use the testnet version of popular wallets to reveal information about the corresponding mainnet nodes (assuming both testnet and mainnet nodes are run on servers with related IP addresses). The main limitation of our technique is the assumption that a user issues multiple transactions during a relatively short time frame through the same node.

There is an inherent trade-off between wallets with centralized and P2P broadcast. Centralized wallets may better protect the user's privacy from external adversaries, but can themselves link users' transactions and correlate them with additional information obtained from the app. Users must also trust centralized wallet providers for availability. Wallets with P2P broadcast eliminate the danger of censorship, denial of service or deanonymization by the wallet provider, but reveal more information to an external observer.

Users of P2P wallets should connect to a trusted full node and avoid sending multiple transactions within a short time frame. The best practices for secure coding are especially relevant for mobile wallets, which run on devices storing lots of personal data. Wallet developers should use as few permissions as possible, open-source the code, provide alternative installation methods (F-Droid, direct APK download), and implement additional network-level measures to prevent traffic analysis.

An attacker may leverage additional information to increase clustering accuracy. For instance, transactions issued by regular users usually contain a small number of inputs and two outputs (destination and change). Transactions with a large number of inputs or outputs are likely to have originated from a node associated with a business (a custodial wallet, an exchange, a mining pool). An attacker trying to deanonymize a regular user can remove clusters with many "enterprise" transactions, making the victim's anonymity set smaller, and use known addresses of various services from sources like [1]. We leave this research direction for future work.

6.1. Ethical considerations

All 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.

7. Related work

Early research on cryptocurrency privacy mostly covered Bitcoin public blockchain graph analysis and proposed techniques such as mixing, where users combine inputs in a joint transaction, making it harder for an adversary to track the flow of coins [8][21]. Gervais et al. [14] analyze the privacy implications of Bloom filters in SPV wallets.

The network-level privacy attacks on cryptocurrency users developed from using a simple "first relayer" heuristic [15] [18] to more sophisticated techniques: fingerprinting by entry set [6] (while abusing the Bitcoin DoS-protection mechanism to prevent the victim from connecting over Tor [7]), exploiting peculiarities in the update mechanism for known address database [16], discovering the network topology from timing analysis [17]. These techniques are being applied to privacy-focused cryptocurrencies as well [20].

Dandelion++ [12] is a message propagation protocol for P2P networks designed to prevent deanonymization attacks by introducing asymmetry in message propagation. In Dandelion++, a message is first propagated though multiple random hops in a linear fashion, and then disseminated with the existing gossip mechanism. The nodes for the first phase are chosen only from outgoing connections. If implemented, it would prevent attacks which rely on saturating free connection slots of remote nodes (including our clustering technique).

8. Conclusion

We studied Android wallets for Bitcoin and the major privacy focused cryptocurrencies. Most wallets do not satisfy our initial privacy criteria. Many wallets obtain dangerous permissions and potentially leak users' private information. Static analysis reveals multiple defects in their source code. P2P wallets do not implement privacy enhancing broadcast mechanisms such as diffusion, which is used by Bitcoin Core.

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.The same set of tools allows an attacker to find out IP addresses of nodes which centralized wallets use for broadcasting transactions.

Privacy-focused cryptocurrencies, while employing sophisticated cryptographic techniques to prevent blockchain analysis, are similar to Bitcoin on the network level. Their lower overall popularity and a smaller anonymity set make them susceptible to network analysis. A smaller number of privacy-friendly wallets exacerbate the threat. We encourage privacy-conscious cryptocurrency users to choose their wallet according to privacy trade-offs. We also suggest wallet developers pay closer attention to privacy on the network level.

[1] Bitcoin block explorer with address grouping and wallet labeling, 2018. `https://www.walletexplorer.com/`.

[2] Bitnodes, 2018. `https://bitnodes.earn.com/`.

[3] Smartdec scanner, 2018. `https://scanner.smartdec.net/`.

[4] 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.

[5] Android. Permissions overview, 2018. `https://developer.android.com/guide/topics/permissions/overview`.

[6] 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`.

[7] 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`.

[8] 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.

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

[10] 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.

[11] Steve Arzt et al. Flowdroid static data flow tracker, 2018. `https://github.com/secure-software-engineering/FlowDroid/`.

[12] 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`.

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

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[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] 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.

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

[23] tyzbit. Samourai wallet needs access to your node's json-rpc port if you're using a trusted node... why? - reddit, 2017. https://www.reddit.com/r/Bitcoin/comments/7qalbo/.