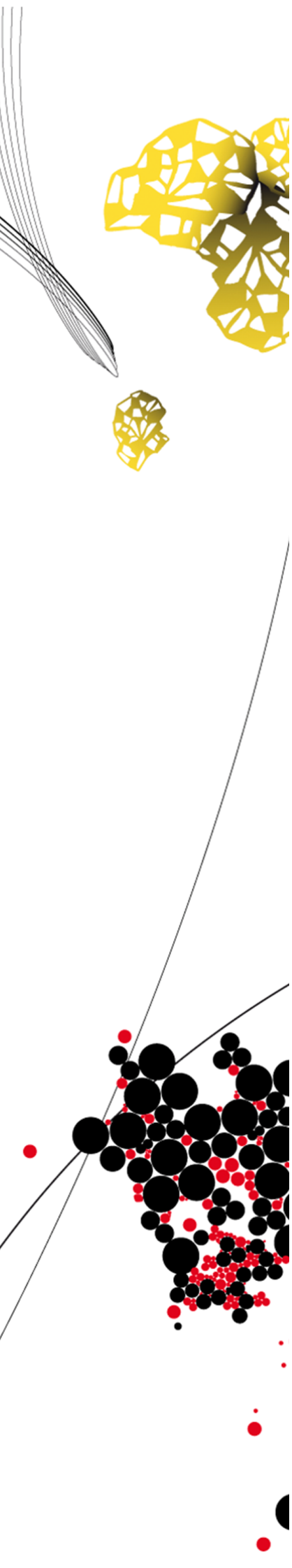

UNIVERSITY OF TWENTE.

Coblue cybersecurity

Gathering Intelligence from the Bitcoin Peer-to-Peer Network

Willem Noort

M.Sc. Thesis
August 2016

Supervisors:
Prof dr. Pieter H. Hartel
Assist.-Prof. Dr. Andreas Peter
Dhr. Remco Bloemen MSc
Dhr. Friso J. Stoffer MSc

Acknowledgments

I would like to thank my family, friends and colleagues at Coblue for their support in the last half year. It would have been impossible to conduct this research without them. I would also like to thank Coblue for allowing me to be a part of the team and providing all required resources as well as a pleasant working environment with lots of laughter and fun.

Finally I would like give a special thanks to all supervisors; Pieter, Andreas, Remco and Friso: thank you for your critical feedback, original insights and ideas, your daily help and your support in writing the report. It is all greatly appreciated.

Abstract

Since the introduction of the Bitcoin cryptographic currency in 2008, several incidents have occurred that involve criminal activity. Investigations of law enforcement agencies are impeded by the decentralized nature of Bitcoin. To support investigations into criminal activity involving Bitcoin, analysis software such as Cointel is developed that combine several approaches proposed in literature to allow Bitcoin users to be tracked. In this work we propose an extension to Cointel to perform analysis on network data that can be obtained by only observing the Bitcoin network. We show that the fraction of transactions that can be associated with the IP address used to spread the transaction increases considerably compared to available literature when other information from Cointel is integrated in the approach. Specifically, we propose and analyze three improvements that combine diverse approaches: firstly, input co-occurrence clustering is used to create groups of transactions that were likely introduced by the same Bitcoin node, secondly we analyze the effect of establishing multiple connection to all Bitcoin nodes, and finally we propose a method to detect nodes that have multiple IP addresses. The main limitation of this work is that only transactions introduced by publicly reachable Bitcoin nodes can currently be deanonymized. We also note that associating an IP address to a transactions is only a starting point for further investigation and that educated Bitcoin users can protect themselves from almost any network related attack.

Contents

Chapter 1

Introduction

Bitcoin is a digital currency introduced by Nakamoto (2008) that distinguishes itself from traditional currencies by not requiring a centralized authority to oversee transactions occurring between the users of the system. Instead, all transactions are made *public* and checked for validity by a network of computers running Bitcoin software. Entities that are using the Bitcoin currency are only known by their 'account numbers' called Bitcoin addresses, but further remain anonymous. The increase of exchange rates of Bitcoin[1] and the increase of available mining power[2] in the last few years indicate an increased popularity of this currency.

That Bitcoin is also being used for illicit activities is evident from several cases. For example, before is was taken down by the FBI in 2013, the *Silk Road anonymous marketplace* mostly known for selling illegal drugs used Bitcoin for payments (Christin, 2013). Additionally, Bitcoin is used as currency in various types of ransomware to receive payment for unlocking encrypted files (Kharraz et al., 2015). The assumed anonymity of users is an important reason for criminals to favor Bitcoin in both examples (Meiklejohn et al., 2013; Sat et al., 2016; Guadamuz and Marsden, 2015). Law enforcements agencies aspiring to persecute criminals using Bitcoin will first need to deanonymize the user that created suspicious transactions.

Numerous approaches have been proposed in the existing literature that can deanonymize or otherwise reduce the privacy of Bitcoin users. Mostly these approaches can be placed in one of the following categories:

1. Blockchain analysis. Approaches in this category use the transactions included in the *blockchain* (the public ledger of Bitcoin) to reduce the privacy of Bitcoin users.

2. Network analysis. Approaches in this category analyze the propagation of transactions over the Bitcoin network or the behavior of the participants of the

[1] See e.g. http://bitcoincharts.com

[2] See e.g. https://blockchain.info/charts/hash-rate

network.

3. Leaks. Approaches in this category usually involve searching for instances where Bitcoin users accidentally or on purpose leak their pseudonyms used in Bitcoin.

The problem of most existing approaches is that most of them do not actually deanonymize many Bitcoin transactions, while others are easy to mitigate. For law enforcement agencies it can therefore be advantageous to compare several (theoretical) approaches to increase to probability of finding some relevant information. The purpose of this study is therefore to find and test combinations of existing approaches that reduce the privacy of Bitcoin users. The research questions are formulated as follows:

> *How can current methods to deanonymize Bitcoin users be combined and expanded to increase the probability that a Bitcoin user can be deanonymized?*

1. What are the existing methods to deanonymize Bitcoin users?
2. Which methods can be combined?
3. What is the practical impact of combining approaches?

Coblue B.V. in Hengelo has provided the assignment and resources for this project. Coblue currently develops Bitcoin analysis software called 'Cointel' to support investigations of law enforcement agencies by providing as much relevant intelligence about Bitcoin addresses and transactions as possible. Currently, Cointel operates by analyzing transactions in the blockchain (the public ledger of Bitcoin) and combining that information with leaks, similar to the work of Spagnuolo et al. (2014). Coblue wishes to extend Cointel with a component that analyses how transactions propagate over the Bitcoin network, even before they are included in the blockchain. For the purpose of performing this research, a proof-of-concept of this networking extension was created, called *Bitsensory*. An important requirement for Bitsensory is that is must be stealthy. To this end, Bitsensory should be protocol compliant and passive attacks are strongly preferred.

The main contribution of this work is a 70% improvement of the approach introduced by Koshy et al. (2014), who links Bitcoin addresses to an IP address that was used to introduce transactions. The Bitcoin network consists of *nodes* that each connect to a few other randomly chosen nodes. If a node creates a new transaction, it is shared with the connected nodes, who in turn do the same until all nodes have learned the transaction. Koshy et al. (2014) connect to all nodes in the network

to discover which node was first responsible for introducing a transaction by simply observing which node first shares a transaction. In practice, this approach fails for most transactions due to randomization of the order in which a new transaction is shared with connected nodes and the small number of transactions that refer a specific Bitcoin address. We propose several improvements that increase the success rate of this approach:

- We do not attempt to associate an IP address to a single Bitcoin address, but to a group of Bitcoin addresses (cluster), each owned by the same entity. For this, existing functionality of Cointel is used.
- We increase the number of connections to all nodes in the network from one to three, to reduce the effect of the randomization when a transaction is shared with connected nodes.
- We introduce a technique to discover which IP addresses are used by the same entity. This is most useful when at the same time a node is reachable from multiple IP addresses, e.g. an IPv4 and an IPv6 address.

This report is the result of a 6 months internship at Coblue. The remainder of this chapter explains some of the fundamentals of Bitcoin and the design of the peer-to-peer network and finally provides some definitions that are extensively used in the remainder of this document. Chapter 2 provides an overview of existing techniques that can be used to deanonymize Bitcoin users. In Chapter 3 we describe the selected combinations and the approach that was followed to test these combinations . In Chapter 4 we introduce *Bitsensory*, our prototype framework that collects and analyzes network data. We describe and validate the proposed improvements of the approach of Koshy et al. (2014) in Chapters 5 to 8, discuss these results in Chapter 9 and the report is concluded in Chapter 10. Additionally, Appendix A includes some other findings and some dead ends that were encountered while conducting this research.

1.1 Bitcoin Fundamentals

In 2008 the *Bitcoin* cryptographic currency was introduced by Nakamoto. At the time, several other digital currencies existed that used cryptography, but Nakamoto introduced a new mechanism to create a network of participants in which no central authorities exist, but where instead all participants agree on the complete history of all transactions.

A system such as Bitcoin can be modeled as a collection of accounts and a list of transactions that transfer value between accounts at a certain point in time. In

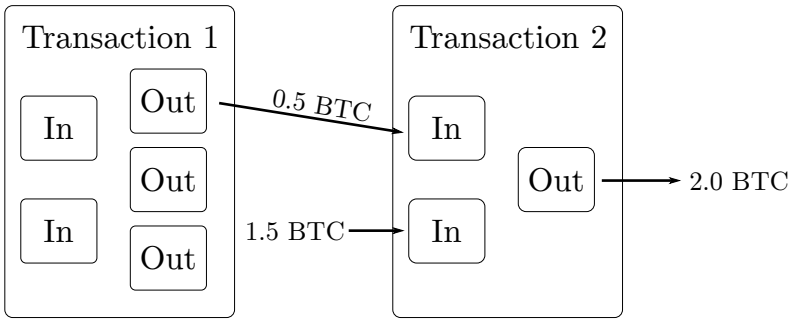

Figure 1.1: Example of how transaction can be linked: Transaction 2 uses an output from Transaction 1 and an output from another transaction to spend 2.0 BTC in total.

Bitcoin an account is a public/private key pair identified by a hash of the public key called a Bitcoin address. To avoid confusion with 'IP address', a Bitcoin address will be indicated with the term '*pseudonym*', as a Bitcoin address acts as a 'pseudonym' of a real Bitcoin user. Initially, a *pseudonum* does not have value on its balance, but new value can be introduced in the system through a process called *mining* (explained later in this section). A transaction is basically a statement that is cryptographically signed by the sending pseudonym declaring another pseudonym the new owner of a certain amount of Bitcoins. New transactions are collected, validated and included in a data structure called a *block* through the process of mining. All blocks reference the previous block, so a chain of blocks emerges called the *blockchain*. The blockchain represents the full state of the Bitcoin system and is stored on all running (full) clients. A pseudonym can be used to sign transactions even if it has insufficient funds, it is sometimes necessary to choose which transaction is valid and which invalid. Intuitively the *oldest* transaction should be considered valid and to this end the position of a transaction (or more precisely the block in which the transaction is included) in the blockchain is used as timestamp.

In the remaining parts of this section several aspects of Bitcoin are explained in further detail:

Transaction The balance of a pseudonym is not stored explicitly, but instead a Bitcoin transaction references previous transaction output(s) in which the value was received. All transaction outputs may be referenced in at most one other transaction to avoid *double spending*. An example of how different transactions can be linked can be found in Figure 1.1.

A transaction output consists of a BTC amount and a *script* that contains the conditions which must be fulfilled for the output amount to be spent, i.e. used as input of another transaction. A common script allows spending the output

only if a transaction that is signed by the private key that belongs to a specified pseudonym, but other script can be made such as *multisig* (x out of y signatures needed).

The sum of output amounts of a transaction should be the same as the sum of inputs. If the sum of inputs exceeds the sum of outputs, the difference is considered a *transaction fee* and may be claimed by the *miner* that included the transaction in the blockchain. Naturally, if the sum of outputs exceeds the sum of inputs, the transaction is rejected.

Block After a transaction is created, it is sent to the other participant in the Bitcoin network. *Miners* include the transaction in a *block*, which consists of a small header and a set of transactions.

In order for the network to accept a block, the hash value (a SHA-256 hash that is applied twice) of the block must start with some zeros as a *proof-of-work* (Back, 2002). Miners accomplish this by varying a nonce field in the block header until they find a block that meets the required *difficulty* (i.e. the number of zeros the hash must start with). The required difficulty is adjusted every 2016 blocks (roughly 14 days) by the network to compensate for an increase or decrease in mining power to ensure that new blocks are found every 10 minutes on average.

Each block starts with a special *coinbase* or *generation transaction* that has no inputs, but outputs 12.5 new Bitcoins (as of July 2016). This is the reward the miner receives for mining the block.

Blockchain All blocks reference the previous block, so a chain of blocks is formed called the *blockchain*. Sometimes multiple valid blocks are found that reference the same previous block: this is called a *fork* and occurs regularly (Decker and Wattenhofer, 2013). When a client notices a fork, it only accepts the chain in which the most work was performed and disregards all other chains.

The above implies that transactions are never definitely committed, as a longer chain that does not contain the transaction can theoretically occur, but the probability of this decreases as a transaction is *confirmed* by more consecutive blocks. Receivers of Bitcoin payments therefore typically wait for several *confirmations* before considering a transaction final.

Validation If a user independently wishes to check his current balance or whether he has received payment, he is required to download and verify the full blockchain, and update it regularly as new blocks are created by miners. This allows the user to get a current list of all *unspent transaction outputs*. The user can now

check the balance of a certain Bitcoin address by calculating the sum of values of all unspent transaction outputs the can be spent by that Bitcoin address.

This process of acquiring a complete list of all unspent transaction outputs is very resource intensive in terms of networking bandwidth, storage capacity and processing power, because it requires the validation of *all* previous transactions. For some devices (e.g. smartphones) it is impossible to perform such a resource intensive task, however such devices can still be capable of performing a less reliable form of transaction validation by considering a transaction valid if it has been included in a block that has been confirmed by at least a certain number of succeeding blocks. This type of validation is called *Simplified Payment Verification* (SPV) and works under the assumption that the minority of miners would not risk including an invalid transaction in their blocks causing it to be rejected by the rest of the network and thereby invalidating the mining reward.

1.2 Bitcoin Network

We have already seen in the previous section Bitcoin participants need to communicate transactions and blocks with each other. More specifically, we define the following tasks:

1. If a user wishes to *spend* Bitcoin, a new transaction is created that must be shared with all miners for inclusion in a new block.

2. If a user wishes to verify whether payment has been received, he must be notified about all blocks in the longest chain and receive new blocks when found.

3. Miners collect transactions and share newly found blocks as quickly as possible to reduce the probability of forks.

To facilitate the above tasks participants connect to each other in compliance with the Bitcoin protocol specification[3]. Currently TCP connections over IPv4, IPv6 and Tor are supported. The Bitcoin network is a peer-to-peer network as no distinction between server and client exists. Specifically, nodes upload and download blocks and transactions to their neighbors.

The remainder of this section describes the Bitcoin network in context of other peer-to-peer networks and elaborates on how transactions and blocks disseminate to all nodes.

[3] see `https://bitcoin.org/en/developer-reference#p2p-network` for a complete overview of the protocol.

	Centralized	**Decentralized**
Structured	-	Tapestry Chord Kademlia
Unstructured	Bittorrent Napster	Freenet Gnutella **Bitcoin**

Table 1.1: Categories of peer-to-peer networks

Peer-to-peer networks

Lua et al. (2005) published a survey on different types of peer-to-peer networks and categorizes the discussed networks according to *structure* and *centralization*. A peer-to-peer network can either be structured or unstructured, depending on whether peers divide responsibility for resources among the active participants. A peer-to-peer network is also either be centralized or decentralized, depending on whether the network depends on a central element responsible for managing resources. See Table 1.1 for a categorization of some well-known peer-to-peer networks.

- In the category of centralized and unstructured peer-to-peer networks, we find file sharing applications such as Napster (Saroiu et al., 2003) and Bittorrent. The main problem fixed in those applications is *bandwidth*: a central registry is only needed for storing meta data of the resources and a list of available peers that share them.

- In the category of decentralized and structured peer-to-peer networks, popular implementations of *distributed hash tables* such as Tapestry and Kademlia are placed. These networks typically solve the problem of *availability* of resources (which are identified by hash value). Responsibility for some resource is shared among a subset of all participants, determined by an algorithm that allows quick location of resources.

- The final category is both decentralized and unstructured. Network in this category could feature *anonymity* such as Freenet. Performance of finding and downloading resources in these network is typically pour compared to other types of networks.

The Bitcoin network can be categorized as a *decentralized*, *unstructured* peer-to-peer network. No centralized register is needed to store meta data of available resources of online nodes, as the availability of new resources (transactions and blocks) and online nodes is communicated to all nodes via *gossip* (explained later in this section). Also, the Bitcoin network is unstructured as all nodes keep a full copy of all transactions and blocks they consider valid and are able to upload them to their neighbors.

One of the challenges of many peer-to-peer networks, Bitcoin included, is to connect to nodes behind a firewall or Network Address Translators (NAT). Mostly these nodes can only create but not accept connections. As a consequence many *unreachable* nodes connect to only a few *reachable* nodes.

Message spreading and peer sampling

Although the Bitcoin network does not protect against attacks against the underlying network, some measures have been taken to protect against malicious peers, by introducing random timeouts for spreading messages. The method for message dissemination and peer sampling in the Bitcoin network is an instance of a *gossip* algorithm. Gossip-based protocols are widely used for dissemination of messages, peer sampling, topology construction, resource management and distributed computation (Kermarrec and van Steen, 2007). Models for gossiping are very similar to disease spreading models and have been thoroughly studied (Haeupler et al., 2012; Boyd et al., 2006). Gossip-based algorithms can be described and compared according to three aspects: *peer selection*, *exchanged data* and *data processing* (Kermarrec and van Steen, 2007).

Peer selection In Bitcoin, all nodes connect to 8 randomly selected nodes to communicate with for the duration of a session. Currently connected nodes (both from incoming and outgoing connections) are called *neighbors* throughout this document.

Data exchanged In Bitcoin, the data that is gossiped can be new transactions or blocks and also addresses of nodes seen online recently. When a node receives a new resource (transaction or block) it does not inform its neighbors immediately. Instead, a neighbor is periodically selected to announce newly learned resources to. This process is called *trickling*. 25% of all received resources, however, are spread to all neighbors immediately. This mechanism provides some privacy to the node that first introduced a new block or transaction to the network, as transactions do not necessarily follow the shortest path between two nodes. The specific implementation of selecting a neighbor when

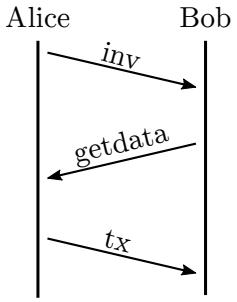

Figure 1.2: Exchange of a transaction between Alice and Bob: Alice announces the availability of a new transaction by its hash value that Bob later requests

trickling resources depends on the used client software:

Bitcoin Core 0.11 and earlier Every 100ms a random connected peer is selected.

Bitcoin Core 0.12 On average every connected peer is selected once every 5 seconds implemented (as an independent Poisson process). The time between random connected peer selection is an exponential distribution with an average of 5 seconds divided by the total number of neighbors.

Data processing Blocks are appended to the local copy of the blockchain and newly received transactions are added to the candidate block by miners or presented to the user if it is considered relevant.

When a node selects a neighbor to share new resources with, the mechanism depicted in Figure 1.2 is used. First an inventory message is sent that contains the hash values of new resources. If the neighbor receives a hash value of an unknown resource, it can then be requested. This is a hybrid between the *push* and *pull* model for message dissemination common in gossiping protocols (Felber et al., 2012). Resources are announced and shared at most once over each connection. Addresses of recently connected peers are also spread through the network similar to the way resources are spread, with two notable differences: addresses of recently connected peers are sent directly instead of being announced first, and nodes select only two of their neighbors to share the information with, instead of all. It therefore takes much longer for an address to reach all nodes than a transaction.

Clients that perform simplified payment validation instead of full validation can limit network traffic by announcing the pseudonyms for which they want to receive the transactions. They do this by sending a *Bloom filter* (1970) to their neighbors. A Bloom filter is a data structure similar to a hash set, solely used to test membership of an item. It features a configurable false-positive rate which is used for in Bitcoin

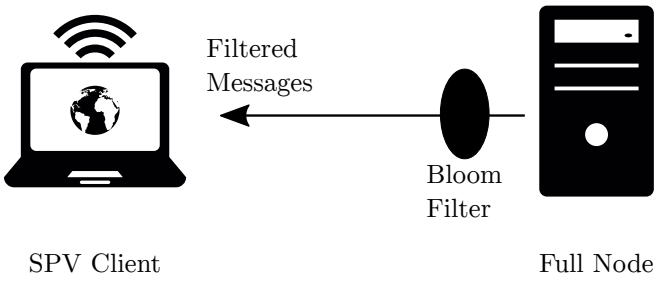

Figure 1.3: Communication adjustments of an SPV Client: A Bitcoin full node applies a Bloom filter to new transaction that are announced to an SPV client. Icons made by Freepik from `http://www.flaticon.com`.

for privacy. A visualization of modified communication to an SPV client can be found in Figure 1.3.

1.3 Definitions

This section clarifies some of the used terminology in this document.

In context of Bitcoin networking a *node* refers to any protocol compliant participant. Several *client* implementations of nodes exist, with Bitcoin Core being most used. Connected nodes of a node are indicated as *neighbors*. The nodes to which an outgoing connection is created are the *entry nodes* ($\subseteq$ *neighbors*) of the node that created the connections.

The node responsible for introducing a transaction is called the *origin node* of that particular transaction. We can observe the origin of a transaction (*observed origin*), which is either a *correct* or an *incorrect observation*.

In this work, a Bitcoin address is indicated as *pseudonym* to avoid confusion with IP address. A transaction is *owned* by the pseudonym(s) used as input. Pseudonyms are in turn owned by the *entity* in control of the private key of the pseudonym. A *cluster* is a group of pseudonyms owned by the same entity. A transactions is also owned by the cluster that included the owning pseudonym.

Deanonymization in this work is linking a transaction, pseudonym or entity to *personally identifiable information* (PII) such as an IP address.

Chapter 2

Deanonymization of Bitcoin Users

As already explained in Section 1.1, all transactions are publicly available in a 'ledger' called the blockchain and all transactions include one or more *pseudonyms* at input and output that are owned by the sender or receiver. The behavior (income and spending) of a pseudonym is therefore completely transparent. The challenge of deanonymizing Bitcoin users is associating pseudonyms to real-life identities such as persons or companies. To this end, various techniques have been proposed in literature to gather additional information about pseudonyms. This could be personally identifiable information (PII), such as online aliases, geographical data or IP addresses used to introduce transactions to the network.

The creation of new pseudonyms is cheap and the developers of Bitcoin advise against using a pseudonym in more than one transaction. We can therefore expect an entity to own many pseudonyms. While the income and spending of individual pseudonyms can simply be discovered by scanning for all related transactions in the blockchain, it only provides a partial view of the transactions of the owner of the pseudonym.

Currently, the available literature on the topic of deanonymization of Bitcoin users utilizes the blockchain (e.g. Meiklejohn et al., 2013) or the peer-to-peer network (e.g. Koshy et al., 2014) as a source of this information. The same research also seems to prove that deanonymization of some Bitcoin users is already possible. In the remainder of this chapter we therefore provide an overview of the existing techniques, which can be subdivided in two main categories:

1. Analysis of the transaction graph. These techniques are mainly used to *cluster* pseudonyms together that are owned by the same entity (Section 2.1).

2. Analysis of traffic from the Bitcoin peer-to-peer network. This can result in associations between nodes in the Bitcoin network and transactions (Section 2.2).

In conformance with the requirement of Coblue to extend Cointel with a network-

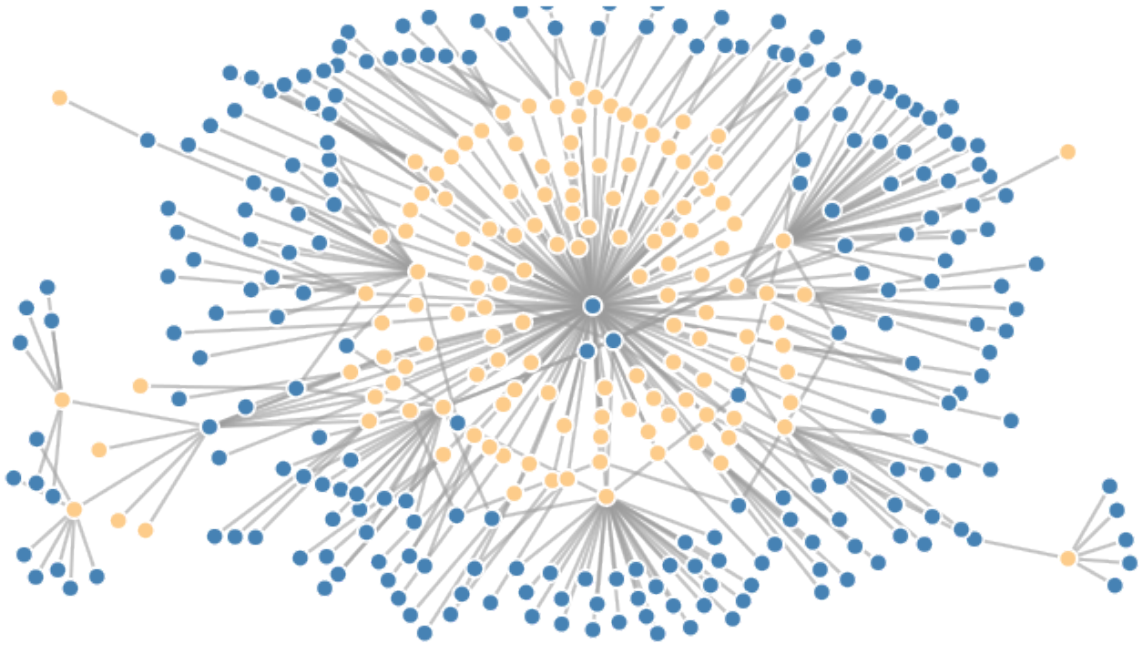

Figure 2.1: Cluster of Bitcoin addresses attributed to WikiLeaks, as created by *Cointel*. A blue dot represents a Bitcoin address and a yellow dot a transaction that links multiple input addresses together.

ing component, some literature that is considered most relevant for this is explored in more detail in Sections 2.2.1 and 2.2.2.

2.1 Transaction Graph Analysis

As the blockchain contains the full history of all transactions it is possible to follow the flow of value, by parsing all transactions included in the blockchain into a directed *transaction graph* (see also Figure 1.1). This transaction graph has been used by several researchers to help deanonymization of entities.

The most important application is 'clustering' of Bitcoin addresses that are owned by the same entity. Clustering heuristics by themselves do not deanonymize an entity, but in combination with other information this results in a powerful attack against the anonymity of Bitcoin users: a cluster of Bitcoin addresses that belong to the same entity can be used to further analyze the behavior for the owning entity. For example, it gives a more complete overview of some person's income and spending. Several heuristics have been proposed that use the transaction graph to cluster addresses that belong to the same entity:

Input co-occurence Nakamoto (2008) notes that if a transaction has multiple inputs, the private keys used to sign the transaction are likely owned by the

same entity, although in theory it is possible that different users provide inputs of a single transaction by sending a partially signed transaction around to all participants until it has been fully signed, and then introduce the transaction to the network. An example of this type of clustering can be found in Figure 2.1, where each blue dot represents a Bitcoin address that is likely owned by WikiLeaks, because they have been used as input of the same transaction (yellow dots).

Change address If a payment is made, it is unlikely that the payer has inputs available that exactly match the amount of Bitcoins that is to be payed. Therefore, many transactions include an extra output that belongs to the payer to transfer the 'change' of the transaction back. The heuristics described below have been proposed to detect which of the outputs of a transaction belongs to the payer and which to the payee.

Meiklejohn et al. (2013) have used the behavior of Bitcoin wallets to generate a *new* 'shadow' addresses, to which the change of a payment can be transferred. So if a transaction has multiple outputs of which only one has not yet been used in the blockchain, the new address belongs to the entity that created the transaction. Another possibility is that one of the output amounts is a rounded number (in either Bitcoin, or when converted to another currency) and one is not, in which case the second output is likely the change.

Many more authors have analyzed the possibility to relate some of the pseudonyms used in a transaction to the same entity, but for the purpose of this research a basic understanding of clustering heuristics suffices.

To counter clustering heuristics completely it is necessary that the inputs and outputs of a transaction remain unrelated. To this end, *mixing protocols* have been proposed such as *Coinjoin* (Meiklejohn and Orlandi, 2015). Mixing involves multiple users providing the inputs and outputs of a single transaction that is signed by all participants, to hide which inputs and outputs are related.

2.2 Network Analysis

This section provides an overview of techniques described in literature to deanonymize Bitcoin users by analyzing network traffic.

The Bitcoin network itself provides no measures to protect the confidentiality or authenticity of the communication between nodes. In a model that assumes an attacker that has full control over the communication channels, such as the Dolev-Yao model (1983), the provenance of messages is leaked, nodes can be isolated from

the rest of the network and an attacker can even control which blocks and transactions a node is aware of (Ali et al., 2015). Instead, all transferred data is considered public, and participants are encouraged to connect through an anonymity network such as Tor when introducing sensitive transactions. The research discussed in this chapter assumes a less powerful attacker, who is unable to control the communication channels of other nodes, but only participates in the network in a protocol compliant manner.

In the remainder of this chapter we describe existing methods to discover which node was responsible for introducing a transaction to the Bitcoin network when this node is either publicly reachable or unreachable (Sections 2.2.1 and 2.2.2 respectively), that apply to most transactions. In the last section (2.2.3) other relevant work is listed that did not achieve deanonymization of Bitcoin users, but did perform analysis of the Bitcoin network that could be useful.

2.2.1 Transaction from reachable nodes

In 2011 security researcher Kaminski introduced the idea that if connected to all Bitcoin nodes, "The first node to inform you of a transaction is the source of it" (2011). This idea has since been used by several other researchers to link transactions to IP addresses. Koshy et al. (2014) created a custom (protocol compliant) Bitcoin client optimized to maintain many connections to other Bitcoin nodes in order to observe the Bitcoin P2P network. During an experiment that lasted for 5 months he observed some different relaying patterns for transactions:

- 91% of the observed transactions were relayed once by multiple nodes (Multi-Relayer, Non-rerelayed Transactions). This is normal behavior expected from Bitcoin clients. The author hypothesized that the first node that relayed a transaction is the owner of the input address of the transaction (Kaminski, 2011).

- 3% of the observed transactions were relayed by only one node (Single-Relayer Transactions). The author hypothesized that if a transaction is only relayed by a single node, this node must be the owner of the input addresses of that transaction.

- 6% of the observed transactions were relayed multiple times (Multi-Relayer, Rerelayed Transactions). As the Bitcoin protocol only allows the sender and receiver to relay a transaction multiple times, the author hypothesized that they relayed the transaction multiple times.

If at least 5 measurements for a single pseudonym are available ('*support count*' of at least 5) and only one IP address is a likely candidate for an association ('*con-*

fidence' of at least 50%), it is considered 'certain'. This way, out of 3.9 million analyzed transactions, several hundred Bitcoin addresses could be associated with the IP address used to introduce the transaction. Most of these high-confidence associations are due to anomalous (Single-Relayer and Multi-Relayer, Rerelayed) transaction patterns. The authors therefore conclude that the followed approach has minimal impact in practice and using an official client and avoid using the same address in multiple transactions is sufficient to avoid detection by this attack. If an official client is used, the second and third relaying patterns are unlikely to occur, and by using the same address in less than 5 transactions, the attacker will never receive sufficient confirmations to make a 'certain' association.

2.2.2 Transactions from unreachable nodes

Building on the research of Koshy et al. (2014), Biryukov et al. (2014) concludes that the impact of the previous approach is further reduced when transactions are introduced by unreachable nodes, and this could even result in false associations. In practice many nodes are unreachable for an attacker, either due to *Network Address Translators* (NATs) or Firewalls. Unreachable nodes can however still create outgoing connections to nodes in the Bitcoin network. Consequently, transactions created by unreachable nodes would not be associated with the correct IP, but with one of the *entry nodes*.

Biryukov et al. (2014) presents a vulnerability in the Bitcoin protocol used to create an attack that specifically targets transactions originating from unreachable nodes. The performance of this attack is evaluated by conducting some experiments in the Bitcoin testing network. This required the development of a custom Bitcoin client that maintained the connections.

The vulnerability is that when reachable and unreachable nodes connect to the network, they will send a message containing their public IP address and a recent timestamp to the entry nodes. The entry nodes will then forward this message to two of his neighbors. It would be possible to connect to all reachable nodes many times to receive such messages with high probability and thus learn the IP address of a newly connecting (unreachable) node and one of his entry nodes as explained in Figure 2.2.

If at least 3 known entry nodes of unreachable nodes are early to announce knowledge of a transaction, then the transaction is created by the unreachable nodes with a high probability, even though the source node itself was not reachable. According to the author, this attack can be used to link 11% of all transactions to an IP address if 50 connections can be established to all reachable nodes, at a

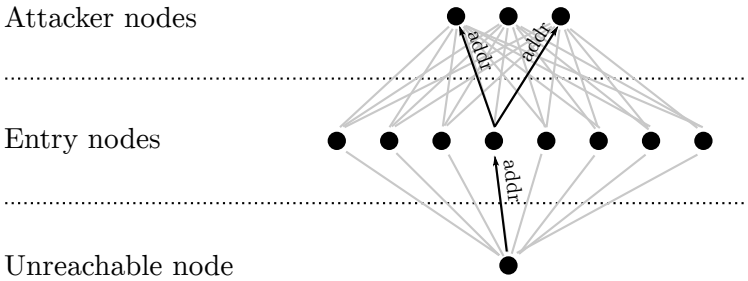

Figure 2.2: Discovering the entry nodes of an unreachable node: the unreachable nodes forwards his IP address to an attacker node via one of his entry nodes. The attacker learns both the IP address of the unreachable nodes and the IP of an entry node with a certain probability. Open connections are depicted in gray. The entry and attacker nodes will also have connections to other nodes, but those are not depicted.

cost of about $1500 per month[1]. A small DOS attack could improve this to 60% of all transactions.

2.2.3 Other work

Gervais et al. (2014) analyzed the implementation of Bloom filters used by Bitcoin clients that perform simplified payment verification (*SPV clients*). It was found that the current implementation of Bloom filters leaks 80% to 100% of the addresses a client is interested in. Biryukov and Pustogarov (2015) have researched deanonymization of Bitcoin users that are connecting to the network via anonymity networks such as Tor[2]. One of the steps of Biryukov and Pustogarov (2015) is a technique to 'fingerprint' clients that are connecting over Tor, so that if a node connects to the Bitcoin network without Tor, it can be identified.

Miller et al. (2015) describe a method to discover the active connections of a Bitcoin node by repeatedly requesting for known addresses. Responses to these requests include for each node a timestamp to indicate its 'freshness'. Miller et al. (2015) observes that these timestamps are updated differently for nodes that maintain a connections and uses this mechanism to infer which nodes are connected. Donet et al. (2014), Decker and Wattenhofer (2013) and Feld et al. (2014) have researched the size, structure and performance of the Bitcoin network and the distribution of nodes around the world. The mechanism to discover active nodes works by repeatedly requesting for new node addresses from nodes that are already known, bootstrapped by some hard-coded nodes addresses.

[1] $1500 is needed to rent the servers needed to establish connections to the other Bitcoin nodes
[2] The Onion Router; see `https://www.torproject.org/`

2.3 Summary

We have seen in this chapter that current literature that describes methods to weaken the privacy of Bitcoin users can be subdivided in transaction graph analysis and network analysis. Methods that analyze the transaction graph use persistent data from the blockchain, while methods that perform network analysis need (many) connections to nodes in the Bitcoin network to gather information.

Most methods described in literature do not actually deanonymize many users. For example, methods that analyze the transaction graph to cluster pseudonyms owned by the same entity provide a more complete view on the spendings and income of a user, but the user remains unidentified. Analysis of network traffic deanonymizes only a small fraction of active addresses (Koshy et al., 2014), or requires so many connections that the attack likely disrupts the network and is easily detectable (Biryukov et al., 2014). Other research does not deanonymize users at all, but only analyses the structure and topology of the Bitcoin network.

It seems that with only a few precautions a Bitoin user can remain anonymous. Mixing services such as Coinjoin (Meiklejohn et al., 2013) can be used to mitigate clustering heuristics and network related attacks can be mitigated by not accepting incoming connections or use a proxy service that hides the used IP address entirely.

When considering network related attacks, the most relevant approaches follow from the intuition of Kaminski (2011) that attempts to find the first node that has learned of a transaction, i.e. the origin node. Both Koshy et al. (2014) and Biryukov et al. (2014) acknowledge that connecting to all nodes and listening for announcements of new transactions is *imprecise*. Koshy et al. (2014) attempts to work around the problem by comparing the results of multiple (related) transactions, only to conclude that the number of transactions that can be related is mostly insufficient. Biryukov et al. (2014) attempts to reduce the source of imprecision for transactions that are introduced by unreachable nodes and increases the number of data points by considering observations from multiple neighbors of the origin.

Chapter 3

Approach

Currently, an observer of the Bitcoin network should be able to observe which (reachable) node first introduced a transaction, which provides at least *some* information that law enforcement agencies can use in an investigation. However, Koshy et al. (2014) showed that this information is unreliable for most 'normal' transactions, as multiple transactions created by the same entity are needed for successful deanonymization, which is not available for most transactions. But even if sufficient related transactions are available, the results are *not necessarily consistent*. For this, we distinguish several problems:

Not connected Biryukov et al. (2014) suggests that most of the created transactions are introduced from unreachable nodes. In this case, an observer is not connected to the origin node of a transaction, which results in wrong and inconsistent observations of the origin of the transaction.

Wrong observation The trickling mechanism used to randomize the propagation of transactions (see Section 1.2) can cause wrong observations if the origin node of a transaction announces the transaction to other nodes first.

Changing IP addresses Transactions that are created by the same entity are not necessarily introduced by the same node. This is especially true if a long time has elapsed between the introduction of both transactions. For example, the entity could have changed the used wallet service, or the IP address of the used node has changed.

Multiple IP addresses or nodes A node could have multiple IP addresses. For example, an IPv6 address and an IPv4 address. Both of these addresses can be observed as origin of a transaction, which results in inconsistent results when the observed origins of multiple transactions are compared.

Note that the above list is comprehensive. Without the problem of wrong observations, the origin node of a single transaction is correctly observed if connected to

the origin (second and first problem respectively). Additionally, if a group of transactions was introduced using one node with a single IP address, then all transactions would have the same origin node from the perspective of an observer. Consequently, in absence of all problems stated above, the observed origins of all transactions in a group would be the same and corresponding to the correct origin node.

An effort to improve the current situation can involve improving the 'support count', which is the likeliness that a transaction can be related to a sufficient number of other transactions, or mitigating (one of) the above problems that can occur when the support count is sufficient.

In order to answer the research questions stated in Chapter 1, we strive to combine different types of approaches. Interestingly, to some extend, the approaches to deanonymize transactions originating from reachable (Koshy et al., 2014) and unreachable (Biryukov et al., 2014) nodes already do this:

- Koshy et al. (2014) use a simple form of transaction clustering to create groups of transactions that are owned by the same entity. Specifically, all transactions that referenced the same input pseudonym were grouped.

- Biryukov et al. (2014) use knowledge about which nodes are connected to deanonymize transactions first observed from the neighbors of the origin instead of the origin itself, which is most useful when the origin itself is not reachable.

Transaction graph analysis from other research could increase the 'support count', while further analysis of the structure of the Bitcoin network can improve the 'confidence'[1] of an IP address when the support count is sufficient. Ideally, all mentioned sources of information would be combined into a single approach to maximize the impact, but for this research this proved infeasible. Partially this is the result from the requirement of stealthiness, that limited the possibility to perform active attacks. See Appendix A for some of the dead ends that were encountered during this research.

In order to provide an answer for the research questions we decided to limit this research to transactions that were introduced by reachable nodes. We improve the existing approach of Koshy et al. in both the support count and the confidence of associations between Bitcoin address and IP by integrating other approaches:

- The **support count** of an association is currently determined by the number of transactions that use the same pseudonym as input. Using clustering heuristics implemented in Cointel, we learn which pseudonyms are owned by the

[1] This term used by Koshy et al. (2014) is somewhat confusing. If for a group of related transactions all IP addresses from which the transactions were first seen are stored, one would expect that the IP address of the Bitcoin node that created the transactions to appear most often. In this context, 'confidence' is the fraction of transactions that was first observed from the most likely candidate IP.

same entity with a high probability, so that the support counts of individual pseudonyms may be combined.

- The **confidence** of an association is currently limited by the possibility that one of the problems occur that are listed at the beginning of this chapter. We propose improvements that limit the possibility of 'wrong observations' and mitigate the wrong observations due to nodes that have multiple IP addresses.

We proceed this work as follows:

1. We introduce *Bitsensory*, the framework that was created to collect and process data from the Bitcoin network as an extension to Cointel (Chapter 4);
2. We introduce and test the impact of the improvements (Chapters 5 to 7) and
3. We test how the proposed improvements affect the original approach of Koshy et al. (2014) (Chapter 8).

Chapter 4

Bitsensory: Extending Cointel with Network Analysis

The intuition of Kaminski that "when connected to all nodes, the first node to inform you of a transaction must be the source of it" 2011 was proven unreliable in the works of Koshy et al. (2014) and Biryukov et al. (2014). However, both approaches still require to connect to all nodes and obverse which node first announces a transaction. Bitcoin Core was not designed to handle such a large number of connections efficiently nor to measure the precise time of arrival of messages. For this reason, the literature described in Section 2.2 either uses a modified version of Bitcoin Core or develops a protocol compliant client from scratch. Because such specialized clients are not yet available for Coblue, it was decided to develop a new specialized client, *Bitsensory*, to collect the required data.

In this chapter we introduce *Bitsensory*, a protocol compliant Bitcoin client specialized in maintaining many connections to Bitcoin nodes. See Figure 4.1 for an overview of the architecture of *Bitsensory*. As can be seen in this figure, *Bitsensory* is a distributed application and split into two separate components that perform different functions:

1. Data Gathering. This *sensor* application is responsible for all interactions with other Bitcoin nodes and forwards data that is considered relevant to the second application.

2. Data Processing. This application analyses received data and presents it to the user.

Bitsensory is intended as an extension of Cointel, so the proposed architecture with Bitsensory included will be discussed in Section 4.1. Next we discuss the features of Bitsensory (Section 4.2) and the separate designs of the sensor and processing applications (Sections 4.3 and 4.4).

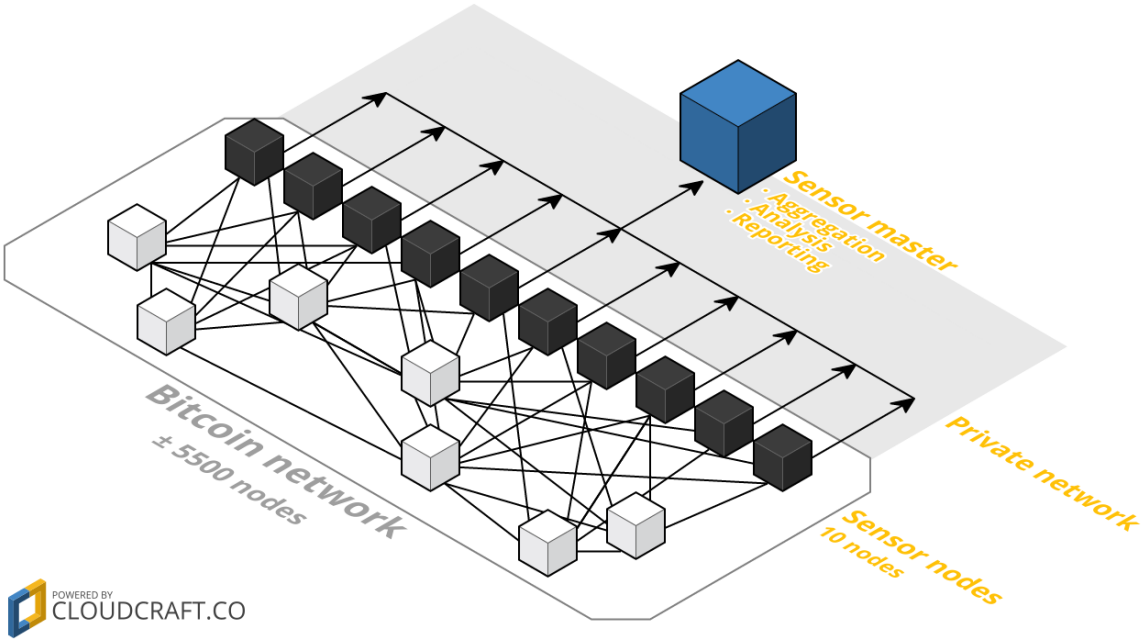

Figure 4.1: Overview of the *Bitsensory* components

4.1 Cointel

Cointel[1] is software developed by Coblue to provide intelligence about transactions useful for law enforcement agencies. Similar to Spagnuolo et al. (2014), Cointel currently consists of two main components:

1. **Scrapers** search The Internet for occurrences of pseudonyms in relation to personally identifiable information (PII) that can link the pseudonym to a real world identity. This can be the case when, for example, a user is accepting Bitcoin donations on their website or a pseudonym is leaked somehow[2].

2. **Clustering** techniques described in Section 2.1 are partially implemented to cluster pseudonyms together that belong the the same entity with high probability. Currently only *input co-occurrence clustering* is implemented to associate pseudonyms that as used as input of the same transaction.

Currently, a component that performs analysis on the Bitcoin network is missing, however the roadmap of Cointel includes an additional component that performs this analysis. See Figure 4.2 for an overview of the envisioned architecture of Cointel with Bitsensory included as a component that performs network analysis.

[1] `http://www.cointel.eu/`

[2] For example, `https://www.walletexplorer.com` maintains a list of Bitcoin addresses linked to an identity

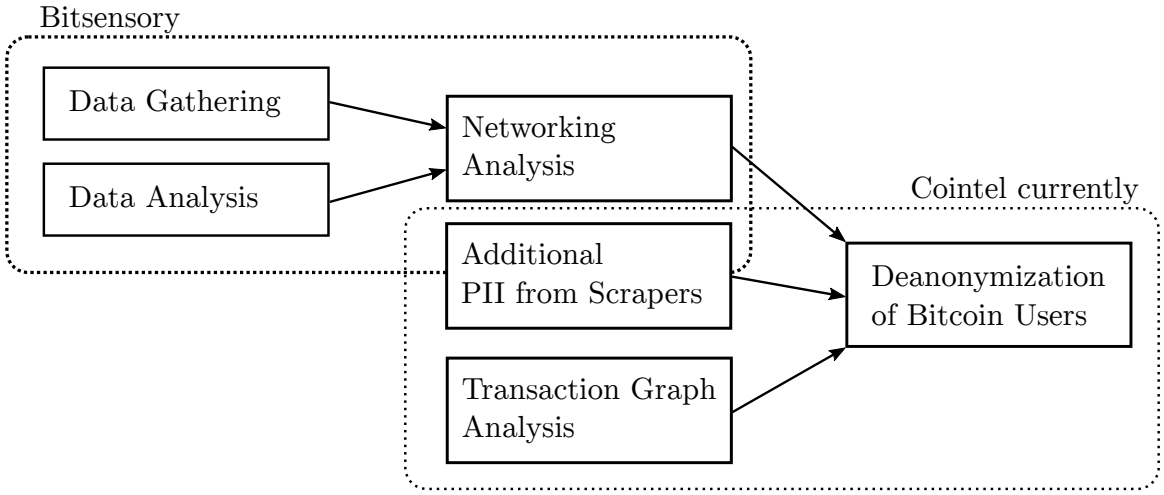

Figure 4.2: Proposed architecture of Cointel when extended with a network analysis component: *Bitsensory*.

4.2 Features

Both Koshy et al. (2014) and Biryukov et al. (2014) require an attacker to learn which nodes first learned about a new transaction, the first node being have created the transaction. However, the exact time at which a node learns about a new transaction is not known. Instead, this time can be approximated by the moment a node first announces the transaction to its neighbors using an inventory message (see Section 1.2 and Figure 1.2).

The primary feature of Bitsensory is therefore to connect to all reachable nodes and log the exact time at which nodes announce new transactions. Additionally, the framework can easily be extended to log other behavior of connected nodes or perform attacks be creating new modules.

Bitsensory finally provides the following notable features:

Distributed The application can run distributed among several machines to allow establishing an arbitrary number of connection to other nodes in the Bitcoin network,

Latency All data received from connected nodes receive a timestamp *before* being processed, for accurate logging of receive time for all observations.

Time synchronization The clocks of the machines are synchronized using NTP, with precision in the lower millisecond. This allows comparing of observations from multiple machines.

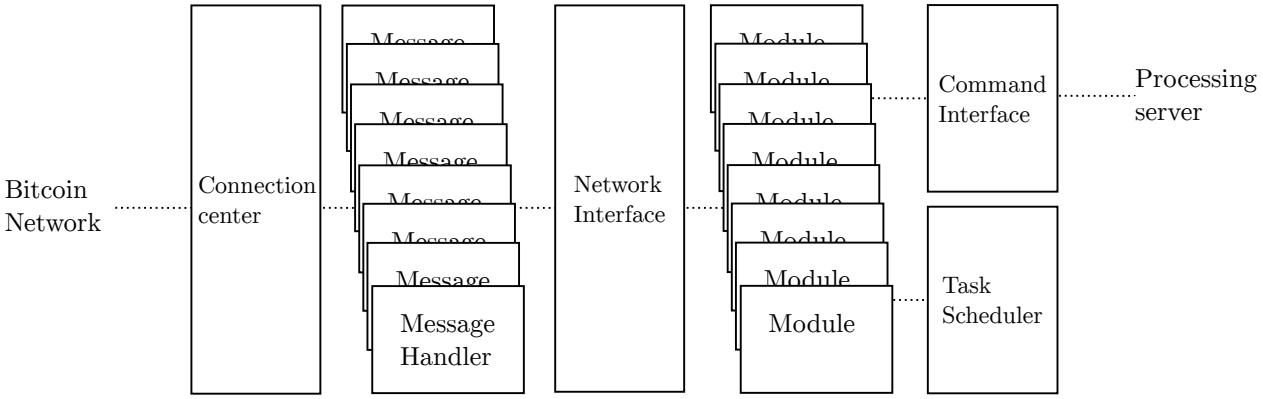

Figure 4.3: Overview of the data gathering component of Bitsensory

4.3 Data gathering

The sensor application is designed to create a single connection to all reachable nodes. It is written in Java and uses the `bitcoinj`[3] library to parse and create messages complaint to the Bitcoin protocol specification. The sensor application follows a modular design and allows plugging and unplugging of modules without restarting or otherwise losing connections. A schematic overview of the application can be found in Figure 4.3 and its components are described below:

Connection Center This component of the sensor is responsible for establishing TCP connections to Bitcoin nodes that are currently reachable. Once a new packet arrives it receives a timestamp and is scheduled for processing. Timing is critical in this component, as delays will influence the accuracy of the timestamps.

Message Handler Multiple message handlers run simultaneously to process all received packets. For all connections the raw data stream is parsed to Bitcoin messages in an accessible format provided by `bitcoinj`.

Network Interface The Network Interface is a *facade*[4] for the entire networking subsystem. It allows registration of callback functions in case some type of message has been received, or the state of a connection has changed and it provides methods to establish connections and send arbitrary Bitcoin messages to active connections.

Command Interface This component provides an interface for communication with

[3] `https://bitcoinj.github.io/`

[4] A single class that provides an easy to use interface to an entire subsystem. See also `https://sourcemaking.com/design_patterns/facade`

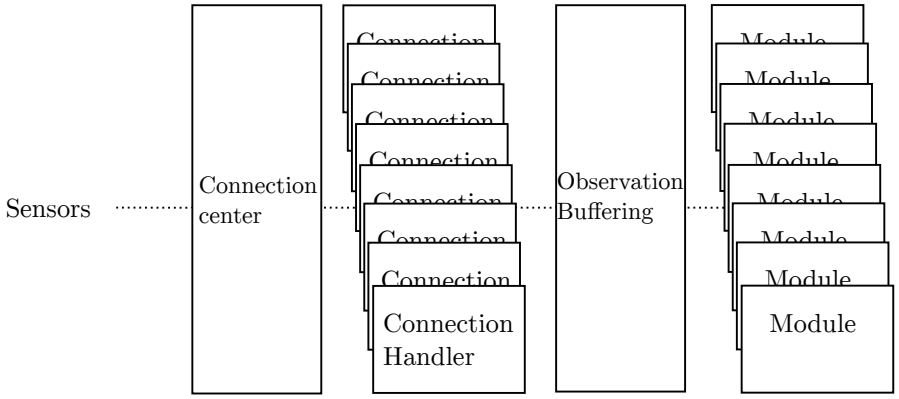

Figure 4.4: Overview of the data processing application

the processing server part of *Bitsensory*. This component also controls the loading and unloading of modules.

Task Scheduler A simple interface that executes a callback function after a predefined timeout or interval.

Modules The data gathering functionality required for *Bitsensory* is implemented as separate modules in the sensor application. A module operates by calling methods and registering callback functions on the three defined interfaces. Modules can be easily implemented for performing attacks and experiments.

Several modules have currently been defined:

`basicprotocol` Ensures that a protocol compliant handshake is performed for all new connections and that the sensor responds correctly to messages received from other nodes.

`interestset` Responsible for discovering online Bitcoin nodes and maintaining connections to all discovered nodes, reconnecting if necessary.

`invlogging` This module listens for `inv`entory messages received from the connected nodes. This information is forwarded to a data processing server.

`txlogging` Requests transaction data if it is still unknown when announced by another node. The transaction data is stored in a database for later use.

4.4 Data processing

The processing application is designed to aggregate all observations of a transaction, perform analysis and archive possibly relevant data. This application is written

in C++ using the Qt5 framework. The processing application follows a modular design al well, where all modules receive a stream of transactions that include relevant data that was gathered by the different sensors. See Figure 4.4 for an overview of the processing application.

The 'Observation Buffering' component of the data processing application introduces a delay of three minutes for all new transactions to collect all observations of the transaction before it is processed in the modules. This ensures that every transaction is processed only once and include all relevant observations.

Currently, the following modules have been defined:

`store` Responsible for storing all data of a transaction to hard disk.

`nodeinfo` Extracts information about Bitcoin nodes from the received transaction observations.

4.5 Deployment

In the final setup of *Bitsensory*, we launched four sensors and one processing server. After running for a few hours, the sensors have discovered all nodes in the Bitcoin network (or at least a similar number to what is reported by several websites that offer real-time statistics on the Bitcoin network), which was between 5200 and 5600 during this research. The processing server stores for each transaction some details about the propagation: the first 500 nodes that announced the transaction, combined with specific timestamps at which the transaction was reported to each connected sensor. This data was compressed and stored, which requires a storage capacity of between 3 and 4 Gigabyte per day.

Chapter 5

Improving Support Counts using Address Clustering

Koshy et al. (2014) have concluded that detecting the first node that propagated a transaction does not work reliable. For some transactions the origin node is observed correctly, but for other transactions a wrong node is detected as origin. In the absence of more information it would therefore be impossible to determine the value of the gathered intelligence.

The above problem can be mitigated if more than one observation can be used, for example from multiple transactions. If for the majority of transactions the same origin was observed, it would be the correct origin for all those transaction. In order to do this, we first need to group transactions that were created by the same entity and therefore (supposedly) introduced using the same node. By analyzing the observed origins of all transactions in a group, it is possible to make more reliable conclusions about which node was the true origin, *or conclude that the origin can not be reliably determined*.

Creation of groups of transactions requires knowledge about which transactions are created by the same entity. In order to do this, Koshy et al. (2014) have used the input pseudonyms of a transaction. Transactions that contain the same input pseudonym, are grouped. This is reliable as those transactions are signed using the same private key under the assumption that entities do not share their private keys.

Koshy et al. (2014) determined that groups in which at least five transactions are included can be used for analysis. Unfortunately, he also concludes that the above method of grouping transactions by input address is ineffective for creating groups of sufficient size (for most transactions). Note that even if transactions are member of a group with sufficient support count, *deanonymization can still fail* due to one of the reasons outlined in Chapter 3.

In Section 2.1 we have seen several techniques aimed at discovering pseudonyms owned by the same entity. We see an opportunity to use these techniques to

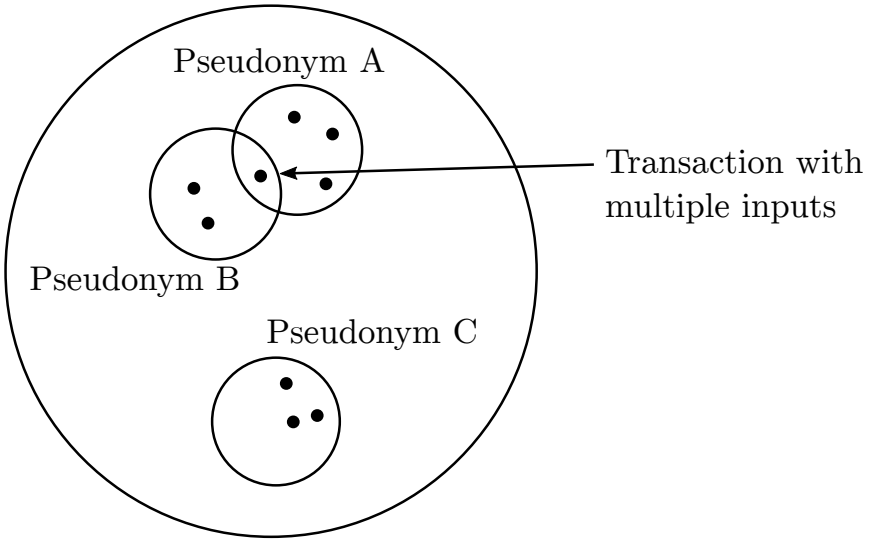

Figure 5.1: Cluster activity.

create larger groups of transactions. Previous groups of transactions can be merged if the pseudonyms used as input belong to the same entity. Of those techniques, *input co-occurrence* clustering is available in Cointel.

In the remainder of this chapter we describe the advantage of integrating clustering heuristics in the existing approach of detecting the origin node of a transaction. We do this by using input co-occurrence clustering to increase the sizes of transaction groups. First we provide a theoretical overview of the expected effect of using clustering heuristics, next the collection of data is described and analyzed to support the theory. Finally, the results are discussed and conclusions are drawn.

5.1 Model

The original approach of Koshy et al. (2014) associated an IP to an individual pseudonym, treating each pseudonym as an entity. In this section we describe the expected effect of using technology in Cointel to cluster pseudonyms and treat each cluster as an entity instead on the support count.

Example Figure 5.1 shows an example of a cluster that consists of 3 pseudonyms A,B and C. All pseudonyms own some transactions (3, 2 and 3 transactions respectively), but none of the pseudonyms have a sufficient support count to allow deanonymization. The transaction that references both pseudonym A and B as input would be ignored by Koshy et al. (2014) as none of the pseudonyms is considered the *exclusive* owner of the transaction. If clustering techniques are allowed that link

the pseudonyms in this example to the same entity, it would then be valid to create a single group of transactions from all transactions owned by this cluster, instead of separate groups each pseudonym. In this example, this would result in a single group with a support count of 9 (which is sufficient), instead of three groups with support counts 3, 2 and 3 respectively, which is insufficient in all cases.

In general, all transactions that are owned by a cluster can be grouped together instead of only the transactions owned by a pseudonym. As additional advantage all transactions can be used instead of only the transactions that reference a single pseudonym at the input, as all transactions are owned by a single cluster, but not all transactions are exclusively owned by pseudonym. Using clusters for grouping transactions instead of addresses will increase the size of transaction groupings as a cluster can contain transactions from multiple addresses, but all transactions owned by an address are also owned by the same cluster.

Temporal separation of groups It would be possible not only to separate transactions according to owning pseudonym or cluster, but also according to time. For example, we can create groups of transactions that were created by a certain entity on a certain day or in a certain month. The most important disadvantage of doing this is the reduced size of groups, caused by the extra constraint. But doing so could also have advantages later in the process of deanonymization, as the behavior of an entitiy is likely more stable in the short term than in a longer term. For example, the IP address of the node that introduces the transactions of an entity can change over time. This could for example happen if the node is located at the home of the owning entity[1].

Experimental data is required to determine the advantage of choosing a larger time interval over small time intervals. If we choose a smaller time interval, more transaction groups can exist that contain transactions that belong to the same entity. Therefore, it is possible to deanonymize the same entity multiple times later in the process and possibly observe changes in the behavior of an entity (e.g, new IP address).

Effectiveness of clustering over time An advantage of using clustering is that the results can improve over time. This happens when after the time of measuring a new transaction is created that combines existing clusters. An example is this can be seen in Figure 5.2, where two clusters are combined due to a new transaction that uses an input pseudonym of both clusters. Within the measuring period, the

[1] Most ISPs either dynamically allocate IP addresses to their customers or share addresses between customers using techniques such as *Carrier grade NATs* (CGN) that could result in an even more dynamic address.

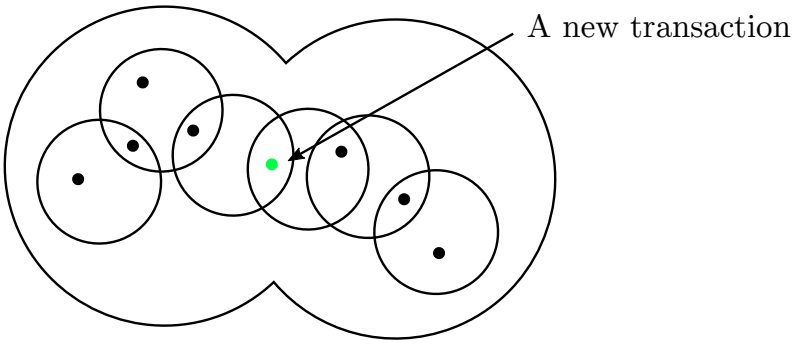

Figure 5.2: Merging of a cluster

Date	March 1 to Junly 31, 2016 (including)
Block heights	400601 - 418877
Transactions	32.7 mln
Unique pseudonyms	34.1 mln
Unique clusters	15.6 mln

Table 5.1: Data collected to analyze influence of clustering techniques on the support count

left cluster in the figure owned four transactions (four leftmost black dots), while the right cluster owned three (three rightmost black dots), so both transactions groups did not have sufficient support count. At any time, a new transaction can appear (green dot) that references a pseudonym of both clusters as input, causing them to merge (i.e. input co-occurrence clustering). As a consequence, the transactions of both clusters can now be added to a single group with sufficient support count of 7 (excluding the green transaction that was created after the measuring period).

In short: the time at which the clustering was performed influences the sizes of the involved clusters and by extend the support counts of transaction groups.

5.2 Experimental results

To experimentally verify the advantage of using clustering heuristics over the method used by Koshy et al. (2014), an experiment was conducted, which is described in the remainder of this section. Furthermore, we explore the influence of different intervals of time on the support counts and the influence of the moment of clustering (i.e. whether clustering becomes more effective over time).

We collected data from real transactions included in the blockchain between March 1 and August 1, 2016. See also Table 5.1. For each day, week and month within this period, we created groups of transactions based on either the input

pseudonym, or the cluster that owned the transaction (based on clustering information that was available at August 4, 2016.). For all these variables for creating groups (starting date, measuring period and cluster or input address) we then calculated the *fraction of transactions that belonged to a group of sufficient support count of five* (further referred to as *success rate*).

This results in a binary experiment for all transactions created between March 1 and August 1, 2016. Success denotes that the transaction was member of a group with at least 4 other transactions and failure denotes that the transaction was member of a group with fewer other transactions.

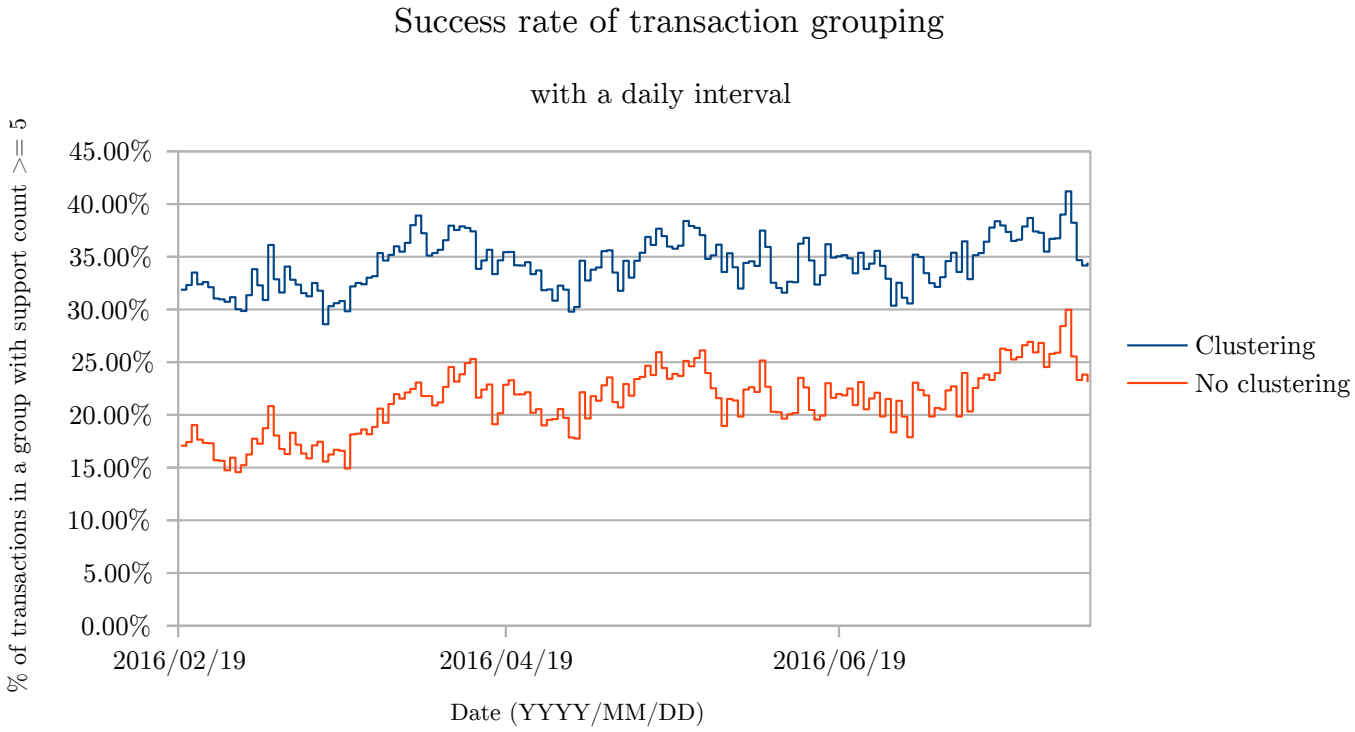

Figure 5.3: Effectiveness of transaction grouping with clustering enabled and disabled

For each day in the measuring period, Figure 5.3 shows the success rate. The blue line ($\mu = 34.3\%$, $\sigma = 2.36\%$) indicates this rate when clustering is enabled, while the red line ($\mu = 21.3\%$, $\sigma = 3.02\%$) shows the rate when clustering is disabled (transactions are only grouped by input pseudonym). This figure shows an average improvement of 13.1 percentage point ($\sigma = 1.43$ pp) when enabling input co-occurrence clustering. The variations in both lines can be explained by considering that the graphs shows the behavior of users, which can change over time.

The results show for all time intervals a *strong correlation* ($\rho = 0.887$) between the clustering and no clustering lines. This makes sense when considering that a cluster consists of one or more pseudonyms. Therefore, the transaction groups created with clustering enabled all consist of one or more groups that would have been created if clustering was disabled. If pseudonyms own more transactions during a certain period (higher success rate when clustering is disabled), then automatically clusters own more transactions during the same period (higher success rate when clustering is enabled).

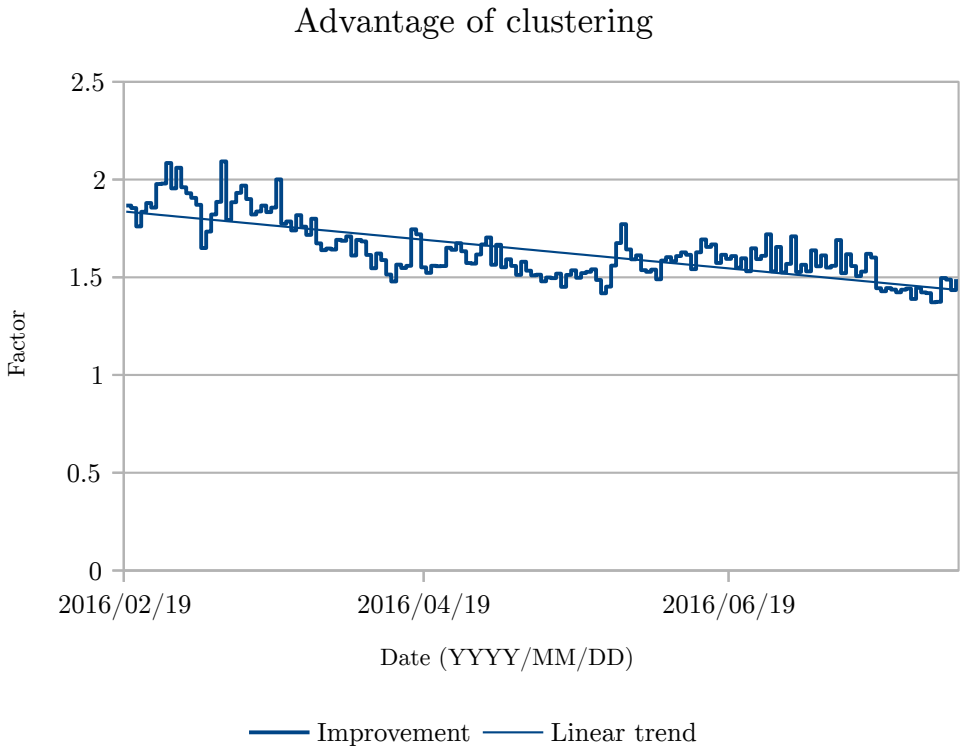

Figure 5.4: Improvement of using clustering over no clustering

The data in Figure 5.3 can be used to determine the improvement of using clustering techniques over the original approach used by Koshy et al. (2014). We express the *advantage* of using input co-occurrence clustering as a ratio between the success rates of the first graph (Figure 5.3) as follows:

$$\frac{\text{success rate clustering}}{\text{success rate no clustering}} \tag{5.1}$$

This is a most intuitive form of expressing the advantage, because the groups created using clustering consist of one or more groups that would have been created without clustering. Figure 5.4 shows the above ratio for all days in the measuring period. The figure shows an improvement with a factor between 1.5 and 2.0. We observe an declining trend of this factor over time, which can be explained by the expected improvement in performance of clustering techniques for older transactions, as explained earlier in this chapter.

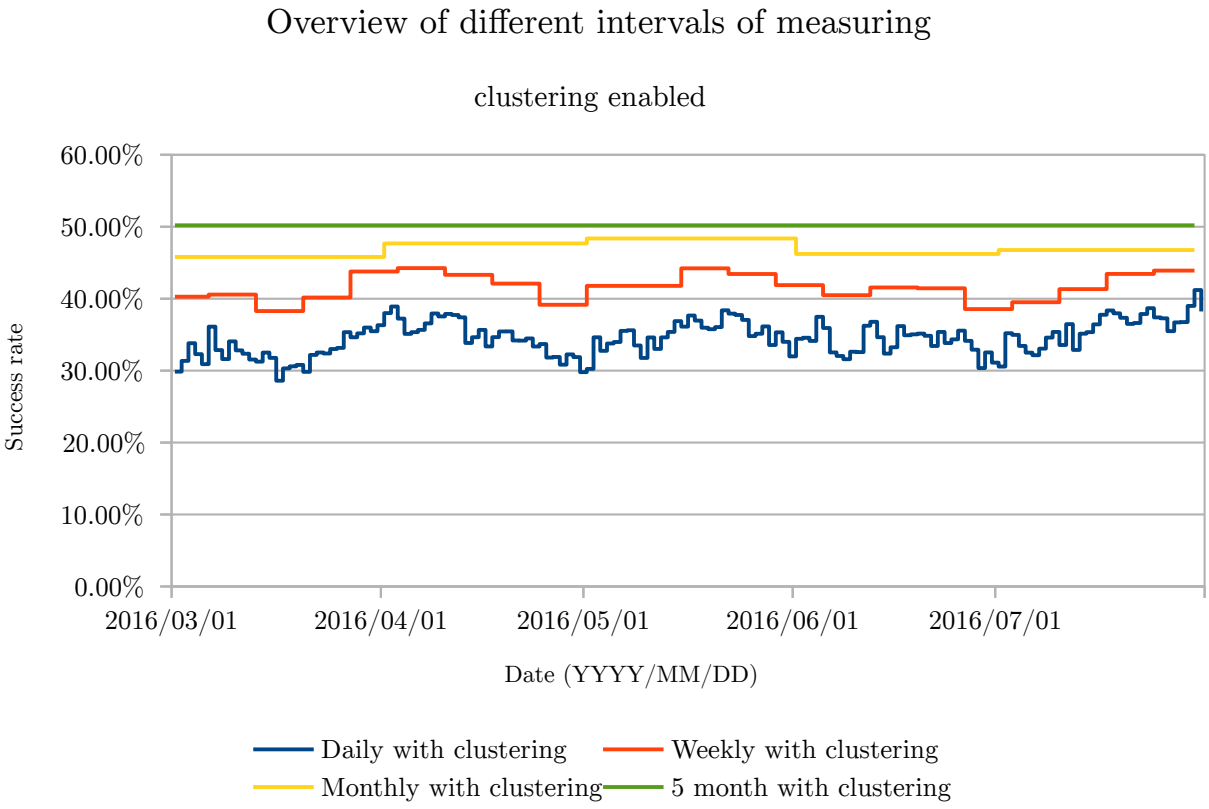

Figure 5.5: Overview of previous results with different time intervals.

Contrary to Koshy et al. (2014), the above results create a new group for each day within the measuring period. When considering other time intervals for which to create new groups, we see the same results as above, but with a higher success rate. Figure 5.5 shows the influence of different time intervals for grouping on the success rate. Choosing a larger interval, consistently results in a higher success rate, to a maximum of more than 50% when considering all transactions created during the 5 month period. For the intervals with clustering enabled, choosing a weekly interval improved the success rate by an average of 7.21 pp ($\sigma = 1.58$ pp), the monthly interval an average improvement of 12.5 pp ($\sigma = 2.24$ pp) and the 5 monthly interval an improvement of 15.9 pp ($\sigma = 2.36$ pp).

This improvement can again be explained from the merging of groups. For example, the activity (transactions) of a certain cluster of a week consists of the activity of that cluster of days in that week.

5.3 Discussion and Conclusions

In this chapter we have shown that input co-occurrence clustering can be used to group transactions together that are created by the same entity. Depending on the used time interval this can result in a success rate of 50% of all transactions being part of a group of sufficient size (five), which possibly allows deanonymization. This is an improvement between 50% and 100% compared to the previous method used by Koshy et al. (2014). When choosing a smaller time interval, the success rate is reduced to 34% when a daily time interval is chosen.

Impact of false clusterings It is possible that clustering techniques yield false results. The clustering heuristic used in this chapter (input co-occurrence clustering) is only reliable under the assumption that a single entity is responsible for creating a transaction. Notable exceptions to this assumption are Coinjoin transactions, that are specifically intended to confuse clustering heuristics. The prevalence of this type of transaction is not investigated in this research, as this does not impact later deanonymization. If unrelated transactions are added to the same group, it will later yield inconclusive results, but no false deanonymizations, as the transactions are introduced using different nodes.

Use of other clustering heuristics Other clustering heuristics are described in Section 2.1 and more clustering heuristics can be discovered in the future. Although only input co-occurrence clustering is used in this chapter, the followed approach can easily be generalized to support any heuristic. Any method that relates different heuristics can be used and will improve the results that were already obtained here.

Chapter 6

Reducing Incorrect Observations with More Connections

We have seen in the Chapter 5 how we can group transactions according to the entity that created the transaction. Using *Bitsensory* introduced in Chapter 4, it is possible to observe the propagation of all transactions in a group over the Bitcoin network. Provided that none of the problems described in Chapter 3 occurs, the observed origin node of all transactions in a group will be the same.

One of the problems described in Chapter 3 is the possibility of 'wrong observations'. This problem can be (at least partially) attributed to the mechanism to disseminate transactions in Bitcoin called trickling, described in Section 1.2. This mechanism randomizes the order in which transactions are shared with neighbors, possibly resulting in a wrong node being observed as origin.

The objective of this chapter is to reduce the possibility of these wrong observations by establishing multiple connections to all nodes in the Bitcoin network instead of only one. In the current literature, only Biryukov et al. (2014) establishes multiple connection to all nodes in the Bitcoin network, but the effect of this is mostly analyzed in relation to the detecting of the entry nodes of unreachable nodes instead of the probability of correctly observing the origin of a transaction.

The remainder of this chapter describes why wrong observations occur and how to mitigate this by establishing more connections to a node. First we provide a theoretical model of correctness of observations, next we perform an experiment to measure the correctness in a practical setting, and finally we draw conclusions.

6.1 Model of Transaction propagation

Types of propagation We have already seen in Section 1.2 that a node chooses to spread all new transactions either by broadcast (25%) or by trickle (75%). To

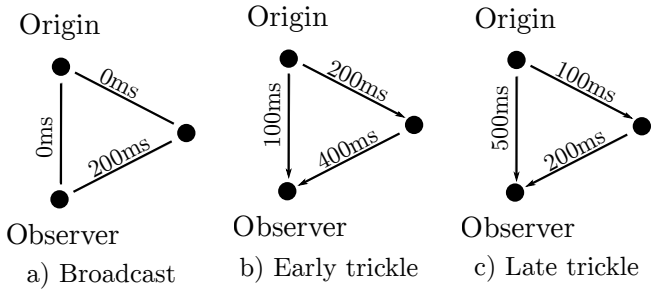

Figure 6.1: Example timings of different paths that a transaction can follow over the network: a broadcast, an *early trickle* and a *late trickle*. Times in this figure are relative to the creation time of the transaction.

avoid easy detection by analysis of broadcasts and trickles, the origin node of a transaction shows the same behavior in this as other nodes. When the propagation of a transaction is observed, the following situation can occur regarding what the origin of the transaction does and what the observer observes (Figure 6.1):

- The origin broadcasts the newly created transaction, so all neighbors simultaneously receive the transaction (apart from networking latency). In this case the number of connections has no influence on the observation, because the transaction is announced simultaneously over all connections. If an observer has multiple connections to the origin, it can observe that the transactions was a broadcast (the transaction was announced at the same time over all connections).

- The origin performs a trickle to randomize spreading of the transaction. Because of this randomization, it is possible that the transaction reaches another node (the 'third node') first that spreads it to the observer *before* the origin does. The observer learns the transaction via a *late trickle* (case c in the figure). In this case, the transactions reaches the observer either via broadcast or trickle, as the 'third node' chooses this independently of the origin.

  Alternatively, the origin performs a trickle to spread its transaction, but choses the observer as (one of) the first neighbors to announce the transaction to: an *early trickle*, which is observed by the observer an a trickle.

The observer is unable to distinguish between broadcasts, early trickles and late trickles. In fact, the observer can only find whether the *observed* origin of a transaction has performed a broadcast or a trickle[1], but the observed origin can broadcast

[1] An observer can distinguish if the first observation was a broadcast or a trickle when multiple connections are established: in case of a broadcast the transaction is announced simultaneously over all connections, but not simultaneously in case of a trickle.

or trickle independent from the behavior of the actual origin of the transaction.

Conservative success rate In this model, the overall success rate of correct conclusions about the origin of a transaction (by definition) depends on the *ratio* between early and late trickles. A conservative approximation of an early trickle would be the probability that the observer was selected first to trickle, where a late trickle is approximated as the probability of not being selected first.

From Section 1.2, we already know that the time between trickles is an independent Poisson process, with the same rate for all neighbors. Therefore each neighbor has the same probability of being selected first as trickle. The probability of a correct observation from a specific origin node can then conservatively be approximated as:

$$0.25 + 0.75 \cdot \frac{\text{Connections to observer}}{\text{Total connections of origin}} \tag{6.1}$$

Here, a broadcast (25% of all transactions) from the origin node will always result in a correct observation, while a trickle (75% of all transactions) results in a correct observation if the observer is trickled first.

Realistic approximation While the above formula gives a certain lower bound on the probability of successfully observing the origin of a transaction, success rates can be better in practice. This is mainly because of two reasons:

- Nodes that receive an announcement of a new transaction do not start propagating immediately. First, some more communication is needed to fully receive the transaction (see Figure 1.2), next the transaction is validated[2] and only thereafter it can be announced to neighbors. As a consequence, if the origin shares a transaction with another node before it is shared with the observer, it still takes some time before this can actually result in late trickles.

- If at some point in time, a transaction has not been announced to the observer yet, but several other nodes already propagate the transaction, there is still a chance that the observer learns the transaction from the correct origin (early trickle) instead of another node (late trickle). This is especially true when considering that some nodes are not reachable and therefore do not directly contribute to late trickles.

The effect of the above reasons on the ratio between early and late trickles depends in practice on various parameters, such as networking delays, validations delays and the fraction of nodes that is reachable. Creating a model with all these

[2]Biryukov et al. (2014) has researched this delay and approximated it to be a typical 100ms.

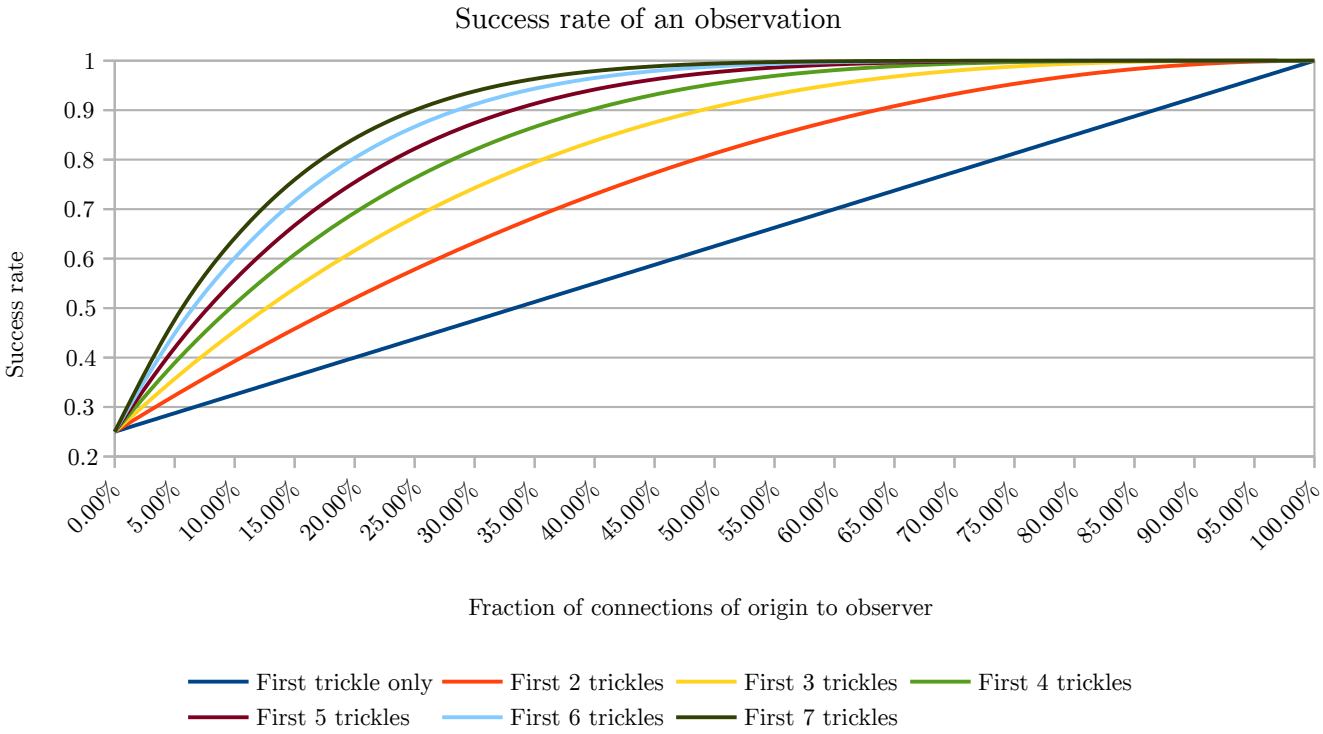

Figure 6.2: Approximation of the probability of a successful observation for different network conditions. Lower latencies will result in success rates closer to the blue line.

parameters would be overly complicated. Instead, we express these networking conditions as an average number of trickles performed by the origin node before another node than the origin announces the transaction to the observer:

$$0.25 + 0.75 \cdot [1 - (1 - x)^n] \tag{6.2}$$

where x is the fraction of connections of the origin controlled by the observer, n is the number of trickles an observer's connections must be among and $(1 - x)^n$ the probability that the observer is *not* among the first n trickles. This equation is a generalization of Eq. 6.1 for $n = 1$. A plot of Eq 6.2 for various values of n is depicted in Figure 6.2.

6.2 Experiment

In the previous section we have seen that the success rate of correctly observing the origin node of a transaction likely depends on both the networking conditions and the fraction of connections of an origin node to the observer.

	Fraction of total transactions	of which trickles	& broadcasts
Order 0	**58%**	62%	38%
Order 1	**22%**	45%	55%
Order 2 (or more)	**20%**	45%	55%

Table 6.1: Accuracy of observations when the connections to the attacker are $\sim 15\%$ of the total number of connections of a node. Order 0 corresponds to correct observations.

Number of connections	1	3
(as percentage of total connection of a node)	~ 5	~ 15
Correct observation rate	38%	58%
Early/late trickle ratio	45/55	20/80

Table 6.2: Overview of results of the connection experiment

We have tested the effect of having more than a single connection to the origin node by launching a new Bitcoin node and use it to create new transactions ($n = 100$). Using *Bitsensory* we then observed the propagation of the created transactions. The results of this experiment can be found in Table 6.1, where an 'order 0' observation indicates a correct conclusion (due to broadcast or early trickle), an 'order 1' observation is when (due to late trickling) a neighbor announced the transaction first and an 'order 2+' indicates that the transaction was first announced by another node (possibly a neighbor of a neighbor, this is also a late trickle).

From this figure and from Table 6.2, we see an overall success rate of 58% and if the observer has about 15% of the connections of the origin (3 out of 22 connections controlled by the observer). When only the observations from a single sensor are considered, the success rate is reduced to 38%. When we insert these results in Figure 6.2, we find that the results are best described by the green line, that assumes networking conditions that allow a correct observation if the observer is among the first four trickles.

6.3 Analysis and Conclusion

In this chapter we have seen how the number of connections to a node influences the probability of successfully determining the origin node of a transaction. Latencies in the Bitcoin network and latencies in nodes cause that an observer not necessarily need to be informed first a new transaction, which reduces the number of connections that is needed.

By introducing 100 transactions to the Bitcoin network while being connected

three times to all reachable nodes, we established that the observer must be among the first four neighbors of the origin that are informed of the new transaction. Although this results were obtained under optimal network conditions (negligible latency between origin node and observer), the same model can be applied to all nodes in the Bitcoin network that use the same mechanism to propagate transactions[3]. In practice this means that when an observer controls 15% of the connections of all nodes, it can correctly determine the origin of 50% of all transactions (this is a conservative approximation[4]).

[3]Broadcast 25% of all transactions and randomly select the order of neighbors for the remaining 75%. This is default for all recent version of Bitcoin Core and forks.

[4]Assuming $n \approx 2.5$ in Eq. 6.2 instead of the measured $n = 4$.

Chapter 7

Detection of Proxies to Cluster Related Nodes

In previous chapters we have seen how clustering heuristics can be applied to improve the number of transactions that can be related to each other (Chapter 5, or how the problem of inaccurate observations can be reduced by establishing multiple connections to all nodes in the Bitcoin network instead of only one (Chapter 6). In this chapter we describe one of the remaining problems that was described in Chapter 3, namely the possibility that multiple IP addresses can be simultaneously responsible for introducing the same transaction.

If a Bitcoin node has several IP addresses, for example in a dual-stack setup, an IPv4 and a one or more IPv6 addresses, all these addresses can be advertised to the network. As a consequence, such nodes are counted multiple times in enumerations that are performed by websites such as `www.nodecounter.com` and `bitnodes.21.co`; the statistics about popularity of different client software can be manipulated[1] and most importantly for this research: transactions that are created by a node that has multiple IP addresses can be first observed via any of the advertised addresses. In other words: the observed origin of transactions introduced by a node with multiple addresses, can be any of these addresses with equal probability[2]. This reduces the confidence of any association that can be made.

Detection of *proxy nodes* can therefore increase the confidence in this specific situation. For example, if the majority of observed origin nodes of transactions in a group are the different IP addresses of a single node, the results would have been inconclusive before, but no longer are if proxies can be detected.

It would have been possible to apply an existing technique to fingerprint nodes

[1] This is relevant in the recent discussion whether to increase the size limitation of blocks, where running alternative clients such as Bitcoin Classic or Bitcoin XT indicates support for a block size increase.

[2] Provided that an equal number of connections is established to all advertised addresses.

across sessions described by Biryukov and Pustogarov (2015) by setting a 'cookie'. However this method is an active attack whereby false node addresses are planted in the target's address manager. Additionally, considerable effort is needed for an attacker to read the cookie out once it has been set. Usage of this fingerprinting technique is also easily detectable and not part of the regular behavior of a Bitcoin client. We therefore decided to search for alternatives that would work without the mentioned disadvantages the approach in Biryukov and Pustogarov (2015).

We have found a simple method to detect proxies that requires simultaneous connections to all candidates, which is descried and evaluated in this chapter. The method to detect proxies functions by creating a 'live' fingerprint of the recent propagation behavior of all *connected* nodes. Nodes that have a sufficiently corresponding fingerprint are considered proxies.

The remainder of this chapter is outlines as follows: first we describe the proposed method of detecting proxies in more detail Next we evaluate the proposed method and some of the parameters that can be chosen. Finally we draw conclusions.

7.1 Method

We have already seen that all nodes randomly determine to perform a broadcast (25% probability) or a trickle (75% probability) for all new transactions they receive. We have also seen in Section 6.1 that this behavior can be detected if more than one connection is established to a node. If a transaction was broadcast by a certain node, all connected sensors would receive the transaction from that node *at roughly the same time*, while a trickle results in a larger variation of received times.

Provided that the decision to perform a broadcast or trickle is indeed made randomly, we can use these measurements as a source of randomness that is unique to each node. Specifically, the entropy of this randomness is can be calculated using a formula of Shannon and Weaver (1949):

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i) \\ &= -(0.25 \cdot \log_2 0.25) - (0.75 \cdot \log_2 0.75) \\ &\approx 0.5 + 0.31 = 0.81 \text{ bit per transaction} \end{aligned}$$

It is also possible to calculate the probability that two different nodes show the same behavior (both broadcasting or both trickling the transaction) for an arbitrary transaction as follows:

$$0.25^2 + 0.75^2 = 0.625$$

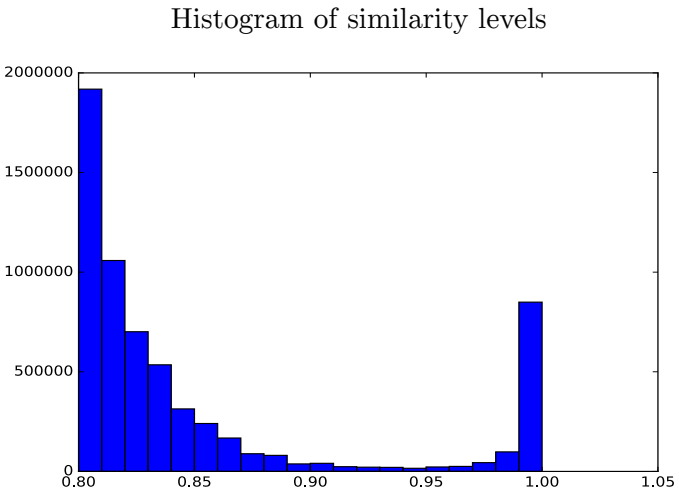

Figure 7.1: Histogram of observed levels of similarity (x-axis) during a week of observing proxies.

The probability that two nodes show the same behavior for 10 randomly chosen transactions is already less than 1% ($0.625^{10} \approx 0.01$). It is therefore very likely that two nodes are actually the proxies of the same node if identical behavior is shown regarding transaction propagation.

7.2 Evaluation

To test the method used to detect proxies in a practical environment, we have implemented an extension of *Bitsensory*. For each connected node, we would store an 512 element shift register. We randomly selected 10% of all processed transactions and pushed the type of observation ('not seen', 'insufficient connections', 'trickle' or 'broadcast') of that transaction in the shift registers of the nodes. Periodically, all registers were compared and if two registers showed a similarity above a certain threshold t, the corresponding nodes were considered proxies.

To empirically determine the correct value of t, we have created a histogram of all similarity levels above 80% that were found during a week of measuring proxies, which is depicted in Figure 7.1. This figure shows that an 80% similarity between fingerprints occurs often, a similarity between 90% and 95% is very uncommon and that a similarity of 98% or more occurs more often again. The increase starting at 95% can not be explained from random broadcast/tricking behavior of nodes and is attributed to the behavior of proxies instead. We therefore set t to 0.95.

7.3 Analysis and Conclusion

Using this approach and parameter $t = 0.95$, we were able to continuously detect approximately 100 nodes that are reachable via multiple IP addresses. This is a relatively small number compared to the total number of connected nodes via IPv4 (~ 4800) and IPv6 (~ 600). With the current state of IPv6 deployment one would expect few nodes to be reachable via IPv6 only, which suggests a high false-positive rate for the followed approach. The false-negative rate of this approach is negligible. This was measured by selecting a random sample of 25 found proxies. For all these proxies was found that both IP addresses of the proxy are owned by the same ISP.

We expect that node clustering only provides a marginal improvement to the original approach by Koshy et al. (2014). If, for example, all 5300[3] nodes would introduce an equal number of transactions, only 1.9% of all transaction could benefit from this method[4].

Also note that this methods only detects whether a single node has multiple IP addresses. If an entity would use multiple nodes (that have different IP addresses) to introduce transactions, it would still result in inconsistent results.

[3] 4800 IPv4 + 600 IPv6 - 100 proxies

[4] Transactions from 100 of the 5300 nodes had inconsistent results due to proxies, which is $\sim 1.9\%$ of all nodes. In this example this also corresponds to 1.9% of al transactions.

Chapter 8

Improvement over Koshy et al. (2014)

Koshy et al. (2014) describe a method that can be used to deanonymize certain Bitcoin users by analyzing the propagation of transactions (see Section 2.2.1). Although the method largely fails for regular transactions and its impact is reduced further by transactions originating from unreachable nodes (Biryukov et al., 2014), improvements may be possible using results from other research.

Additionally, the data used by Koshy et al. was collected between July 2012 and January 2013. Between then and now (June 2016) much has changed: the size of the blockchain has increased from under 5GB to over 70GB and the Bitcoin Core reference implementation has undergone 5 major releases that have implemented several *Bitcoin Improvement Proposals* (BIPs) and *soft forks*[1]. These differences indicate that repeating the experiment could yield different results now.

Furthermore, we would like to compare the results of Koshy et al. (2014) for regular transactions[2] with the improvements explained in the previous chapters enabled:

1. Bitsensory establishes 4 connections to all reachable nodes in the Bitcoin network to determine the origin node of all transaction submitted within a certain period. As concluded in Chapter 6, 15% to 20% of the connections of a node most be controlled by the observer to allow a conservative 50% of all transactions to be linked to the correct origin. We therefore conservatively expect that 4 connections is sufficient for nodes that have 27 neighbors.

2. Using the input co-occurrence clustering heuristic that is available in Cointel, transactions are grouped that are created by the same entity. Compared to the approach that was followed by Koshy et al., Chapter 5 shows that 50% to 100% more transactions are in a group of sufficient size to allow deanonymization.

[1] A forward incompatible protocol update: previously valid behavior becomes invalid after the update. The opposite is a *hard fork* which is a backward incompatible protocol update: previously invalid behavior becomes valid.

[2] Koshy et al. (2014) has also deanonymized transactions by detecting some anomalies in transaction propagation, which was ignored during this research.

IP address	count
54.236.xxx.xxx	5
4.15.xxx.xxx	2
46.166.xxx.xxx	1
5.135.xxx.xxx	1
85.25.xxx.xxx	1

Table 8.1: Example of candidate origin nodes of a transaction group containing 10 transactions.

3. If a node has multiple IP addresses, they can be detected according to the method described in Chapter 7. This prevents inconsistent observations of the origin of transactions introduced by that node.

The remainder of this chapter is structured as follows: Section 8.1 describes how the approach of Koshy et al. (2014) is adjusted to integrate the above improvements, and Section 8.2 compares the impact of the original and improved approach.

8.1 Overview of the improved approach

Using the improvements described in the previous chapters, the approach of Koshy et al. (2014) can be improved, which is described in this section. Note that we only attempt to deanonymize 'regular' transactions described as 'Multi-Relayer, Non-rerelayed transactions' in Section 2.2.1.

The remainder of this sections is a step-by-step description of the improved approach:

Step 1: Data gathering Using Bitsensory we establish 4 connections to all reachable nodes in the Bitcoin network. This will improve the reliability of all observations compared to the single connection that was established in the original approach. To simulate a situation in which only one connection was established to all nodes, we randomly select one observation per transaction per node when these results are compared to Koshy et al. (2014).

Step 2: Grouping Clustering techniques are used to determine the owning entity of the received transactions, as described in Chapter 5. All transactions are grouped according to owning entity. Groups of transactions that have an insufficient size (support count lower than 5) are discarded.

Step 3: Determining confidence Using the data that was gathered by Bitsensory, the origin nodes of the transactions are in a group are determined and counted

Date	June 1st to June 7th (including)
Number of transactions	1.4 million
Addresses referenced	1.8 million
Number of clusters	800,000

Table 8.2: Data used to test the impact of the proposed improvements

(see Table 8.1 for a real example). If one of the nodes is responsible for introducing at least 50% of all transactions in the group, the association between entity and node is considered 'certain'[3]. In this step we consider IP addresses that belong to the same node as a single IP address (Section 7).

8.2 Results

In this section real data from the Bitcoin network is analyzed to test how many identities can be deanonymized using the original approach of Koshy et al. (2014) and compare this to the improved approach as described in the previous section.

The data that was used for this analysis includes the observations of all transactions included in blocks between July 1st and 7th 2016 (See Table 8.2 of an overview of the used data). The results of this analysis can be found in Table 8.3. For both the original and the improved approach, we show the number of pseudonyms and clusters to which an IP address could be associated.

In several cases multiple pseudonyms owned by the entity could be linked to an IP address. Mostly these pseudonyms were linked to the same IP, but for 6 clusters not all pseudonyms were linked to the same IP and not included in these results.

For both approaches we have also calculated the fraction of transactions that could be associated to a deanonymized pseudonym or cluster. For the original approach we considered transactions deanonymized that used a deanonymized pseudonym as (only) input[4], and for the improved approach we considered all transactions that were owned by a deanonymized cluster.

For the improved approach, the IP address that was linked to a cluster was found either by grouping transactions by input address (original) or by grouping all transactions from that cluster.

Overall, we observe an improvement of 70% of deanonymized clusters when all

[3]This is the same as in Koshy et al. (2014).

[4]By only considering transactions deanonymized that reference a Bitcoin address to which an IP could be related, the original results of Koshy et al. (2014) are best represented for this new dataset. It would however be possible to apply clustering *afterwards* which would increase the fraction of deanonymized transactions to 3.1%.

	Original		Improved	
pseudonyms deanonymized	340	$\longrightarrow$	485	
association by considering complete clusters	0		419	
association by considering pseudonyms	272		347	
overlap	0		304	-
Clusters deanonymized	272	$\longrightarrow$	**462**	
Fraction of transactions deanonymized	0.6%	$\longrightarrow$	**4.1**%	

Table 8.3: Results overview

improvements are enabled compared to the original approach and an almost 7 times increase in number of deanonymized transactions.

8.3 Analysis and Conclusion

This chapter show that the results from earlier chapters can be successfully applied to improve an existing approach of Koshy et al. (2014) to deanonymize entities. In this example we have shown how 70% more entities could be deanonymized and that almost 7 times more transactions could be attributed to the denonymized entities.

An important limitation however, is that this experiment could only be conducted using data of *one week*. Consequently, it is impossible to analyze if these results fluctuate over time.

Chapter 9

Discussion

9.1 Impact

Previous chapters have described an improvement of the approach of Koshy et al. (2014) that allows a powerful observer to link some transactions to an IP address. This section places this result in the context of the activity of the full Bitcoin network and compares it to other approaches.

The number of clusters that was deanonymized during the experiment is negligible compared to the total number of active clusters (462 out of 800,000), but if only clusters are considered that have a sufficient support count (≥ 5), this fraction improves to 4% (462 out of 11519). This and the fact that 4.1% of all transactions was owned by an deanonymized cluster suggests that *active* Bitcoin users can be deanonymized with a propability up to 4%.

These results further improve over time if clustering heuristics increase the support count. If, for example, two clusters are merged after some time because a new transaction links them together, the support count of both clusters can be added in retrospect. Adding new clustering heuristics will have a similar effect. Unfortunately, these effects could not be reliably measured within the limited time of this research.

If the improved approach of Koshy et al. (2014) is compared to Biryukov et al. (2014) (Section 2.2.2), we find that the improved approach deanonymizes less transactions (4% instead of 11%), but at a much lower cost (4 connections to all reachable nodes instead of 50).

9.2 Further steps

The most valuable information that can be obtained by analyzing Bitcoin network data are IP addresses from which transactions were introduced. After an IP address has been found, further steps need to be taken to relate this IP address to a person

or an organization (something a law enforcement agency should be able to do). The resulting identity could then either be the owner of a pseudonym, or a third party that is trusted, e.g. a wallet service. Consequently, finding the IP address that was responsible for introducing suspicious transactions is only a first step in an investigation.

9.3 Mitigating the improved approach

For an educated user it is simple to mitigate all network related attacks by *not being part of the Bitcoin network* when submitting transactions. As an alternative to running a Bitcoin client, several online wallets provide a transaction submission form[1], that can be used if these wallets can be trusted to protect the privacy of the submitter. Also, users could connect to nodes that are reachable as a hidden service in the Tor network[2], to anonymously post transactions.

However, both these methods introduce other weaknesses: in case of a submission form, the other party must be trusted to protect the privacy of the user and the anonymity provided by Tor is not perfect (e.g. Shmatikov and Wang, 2006; Danezis, 2003; Zhu et al., 2010). Therefore, completely mitigating risks related to introducing sensitive transactions may not be possible.

Users can verify that a transaction was successfully introduced to the network by running a regular Bitcoin client to check if transaction have been included in the blockchain, without risk of deanonymization.

9.4 Dependency on the propagation mechanism

As a final remark, we would like to point out that both the model to calculate the required number of connections (Chapter 6) and the method to find proxies (Chapter 7) depend on a specific mechanism that to propagate transactions. Two aspects play an important role in this research:

1. **Broadcasts and trickles with a 25% and 75% probability respectively**. For the method to detect proxies, it is only important that different methods to propagate transactions can be distinguished by an observer. It would be beneficial for the method to know the probability that each method occurs in order to

[1] e.g. `https://blockr.io/tx/push`, `https://live.blockcypher.com/btc/pushtx/` or `https://blockchain.info/nl/pushtx`

[2] Bitcoin Core has extensive support for Tor; see `https://github.com/bitcoin/bitcoin/blob/master/doc/tor.md` and `https://bitnodes.21.co/dashboard/?days=90#nodes` for the number of hidden services.

calculate how much entropy is collected for each transaction, but this is not a requirement as long as a lower bound on the entropy per transaction can be determined. A possible change in probabilities of broadcasts or trickles does not fundamentally affect the approach. Entirely disabling either broadcasts or trickles is unlikely to occur as both mechanisms are needed to provide quick propagation of transactions on the one side, and privacy to the other side.

2. **Random selection of neighbors when trickling**. The model that describes how many connections are needed currently assumes that each neighbor has an equal probability of being chosen next to share a transaction. Another distribution of this would not affect the model much as long as the probability of the observer being chosen next can be calculated.

In short, we do not expect that the propagation behavior will change fundamentally, but if it does, the improvements described in Chapters 6 and 7 can most likely be easily adjusted.

Chapter 10

Conclusion

Combining different attacks on the privacy of Bitcoin users is an important step for law enforcement agencies to increase the probability of deanonymizing suspicious transactions. This work shows how some approaches can be combined to create a *passive attack* that is able to deanonymize almost 7 times more transactions than a similar attack that is described in literature.

In this work we have shown how a proposal to deanonymize Bitcoin users, namely by detecting from which IP address a transaction was introduced, can be improved. This attack required an observer to connect to all Bitcoin nodes and observe for each transaction which node was first to start sharing that transaction with other participants in the network. In practice this approach was found inaccurate (Koshy et al., 2014; Biryukov et al., 2014) for several reasons. Firstly, in practice it is not possible to connect to all Bitcoin nodes, e.g. not all nodes accept incoming connections. Secondly, due to randomization in the propagation of transactions, it is possible that the first observation is not from the origin of the transaction. The observations of multiple transactions that were created by the same entity can be combined and analyzed to reduce the impact of the above problems, but is hindered by entities that switch nodes and nodes that are reachable via multiple IP addresses.

We have systematically shown that some of these problems can be reduced by combining several approaches. To this end we developed *Bitsensory*, a piece of software that can observe the propagation of transactions over the Bitcoin network. Firstly, we showed that establishing multiple connections to all nodes positively influences the quality of observations. Secondly we showed that transaction graph analysis improves the number of transactions of the same entity that can be compared. Finally we showed how nodes with multiple IP addresses can be detected.

Not all problems have been reduced in this (and other) work, but Bitcoin users may expect that the used IP address could be discovered by an observer. This may be valuable information for law enforcement agencies, although users can still hide their real IP address by using Tor or a wallet service.

10.1 Future work

Unfortunately, we were unable to integrate some of the research of Biryukov et al. (2014) in our approach. This research describes how transactions introduced by unreachable nodes can be deanonymized by ascertaining the neighbors of those nodes. The main reason was that this attack is not stealthy. Some research has already been conducted to make this approach more suitable, which can be found in Appendix A.2. Another method to deanonymize transaction from unreachable nodes, can be found by performing a Sybil attack on the Bitcoin network. By dominating the set of reachable nodes (currently only about 5500), an attacker could convince unreachable nodes to connect to the attacker, instead of the other way around. Such an attack also requires more research.

Finally some of the results in this work can be investigated further. Figure 5.3, shows some interesting (periodic) fluctuations in the success rates that could not be entirely explained. Also, the results when detecting proxies suggest that not all proxies have been found yet by the described method. Choosing different parameters, or other improvements can result in the detection of more proxies.

Bibliography

Ali, S., Clarke, D., and McCorry, P. (2015). Bitcoin: Perils of an unregulated global p2p currency. In *Proceedings of the 23rd International Workshop on Security Protocols*, pages 283–293.

Back, A. (2002). Hashcash - a denial of service counter-measure. retrieved from `http://www.hashcash.org/papers/hashcash.pdf` March 2016.

Biryukov, A., Khovratovich, D., and Pustogarov, I. (2014). Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM.

Biryukov, A. and Pustogarov, I. (2015). Bitcoin over tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134. IEEE.

Bloom, B. (1970). Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized gossip algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530.

Christin, N. (2013). Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224. ACM.

Danezis, G. (2003). Statistical disclosure attacks: Traffic confirmation in open environments. *IFIP Advances in Information and Communication Technology*, 122:421–426.

Decker, C. and Wattenhofer, R. (2013). Information propagation in the bitcoin network. In *2013 IEEE Thirteenth International Conference on Peer-to-Peer Computing (P2P)*, pages 1–10. IEEE.

Dolev, D. and Yao, A. (1983). On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208.

Donet, J., Prez-Sol, C., and Herrera-Joancomart, J. (2014). The bitcoin p2p network. In *International Conference on Financial Cryptography and Data Security*, pages 87–102. Springer.

Felber, P., Kermarrec, A.-M., Leonini, L., Rivire, E., and Voulgaris, S. (2012). Pulp: An adaptive gossip-based dissemination protocol for multi-source message streams. *Peer-to-Peer Networking and Applications*, 5(1):74–91.

Feld, S., Schnfeld, M., and Werner, M. (2014). Analyzing the deployment of bitcoin's P2P network under an as-level perspective. *Procedia Computer Science*, 32:1121 – 1126.

Gervais, A., Karame, G., Gruber, D., and Capkun, S. (2014). 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.

Guadamuz, A. and Marsden, C. (2015). Blockchains and bitcoin: Regulatory responses to cryptocurrencies. *First Monday*, 20(12).

Haeupler, B., Pandurangan, G., Peleg, D., Rajaraman, R., and Sun, Z. (2012). Discovery through gossip. In *Proceedings of the Twenty-fourth Annual ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '12, pages 140–149. ACM.

Kaminski, D. (2011). Black ops of tcp/ip. `http://www.slideshare.net/dakami/black-ops-of-tcpip-2011-black-hat-usa-2011`. Accessed: March 2016.

Kermarrec, A.-M. and van Steen, M. (2007). Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*, 41(5):2–7.

Kharraz, A., Robertson, W., Balzarotti, D., Bilge, L., and Kirda, E. (2015). Cutting the gordian knot: A look under the hood of ransomware attacks. In *Proceedings of 12th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer.

Koshy, P., Koshy, D., and McDaniel, P. (2014). An analysis of anonymity in bitcoin using p2p network traffic. In *Proceedings of the 17th Conference on Financial Cryptography and Data Security*, pages 469–485.

Lua, E., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. (2005). A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys and Tutorials*, 7(2):72–93.

Meiklejohn, S. and Orlandi, C. (2015). Privacy-enhancing overlays in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 127–141. Springer.

Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G. M., and Savage, S. (2013). A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 127–140. ACM.

Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., and Bhattacharjee, B. (2015). Discovering bitcoins public topology and influential nodes.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.

Saroiu, S., Gummadi, K., and Gribble, S. (2003). Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Systems*, 9(2):170–184.

Sat, D., Krylov, G., Evgenyevich, K., Bezverbnyi, Kasatkin, A., and Kornev, I. (2016). Investigation of money laundering methods through cryptocurrency. *Journal of Theoretical and Applied Information Technology*, 83(2):244–254.

Shannon, C. and Weaver, W. (1949). *The Matematical Theory of Communication*. University of Illinois Press.

Shmatikov, V. and Wang, M. . (2006). Timing analysis in low-latency mix networks: Attacks and defenses. In *Computer Security ESORICS 2006*, pages 18–33. Springer.

Spagnuolo, M., Maggi, F., and Zanero, S. (2014). Bitiodine: Extracting intelligence from the bitcoin network. In *Proceedings of the 18th Conference on Financial Cryptography and Data Security*, pages 457–468. Springer Berlin Heidelberg.

Zhu, Y., Fu, X., Graham, B., Bettati, R., and Zhao, W. (2010). Correlation-based traffic analysis attacks on anonymity networks. *IEEE Transactions on Parallel and Distributed Systems*, 21(7):954–967.

Appendix A

Other findings and Dead ends

While conducting this research we have encountered several dead ends, an overview of which is provided in this appendix.

A.1 Bloom filters

As already mentioned in Section 2.2.3, Gervais et al. explored the privacy provisions of Bloom filters in the context of Bitcoin SPV clients. Some vulnerabilities were found: SPV clients usually reinitialize their Bloom filter when a new session starts, so different Bloom filters may be acquired. Depending on how many Bloom filter were acquired, the following was discovered:

- If a single Bloom filter was received to which fewer than 20 pseudonyms were added, about 80% of these pseudonyms can be recovered;
- If multiple Bloom filters containing the same pseudonyms were received, (almost) all pseudonyms can be recovered, irrespective of how many were added to the filter, by looking at pseudonyms that reported membership in all received Bloom filters.

The use of this attack in practice is not discussed in Gervais et al. (2014). It therefore remained unclear what the exact cost and impact of performing the attack is. If, for example, an SPV client would accept incoming connections, the cost will be low as an attacker can connect to the SPV client to acquire a Bloom filter. If the client however does not accept incoming connections, the cost will be much higher, as an attacker would then needs to run many reachable Bitcoin nodes and hope an SPV client will connect to one of them (a *sybil attack*). Contrary to what was indicated by Gervais et al. (2014), some reachable nodes actually presented a Bloom filter when Bitsensory attempted to connect. Further research indicated that this filter

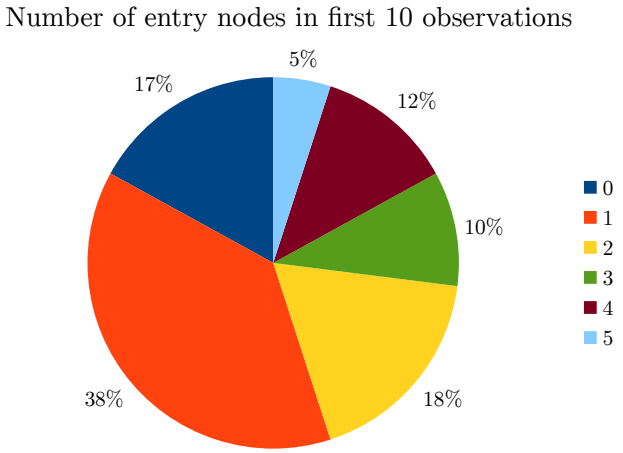

Figure A.1: Number of entry nodes in top 10 of early observations

would match on any pseudonym and is only used by the Bitcoin XT client to speed up propagation of blocks[1].

A.2 Transactions from unreachable nodes

As the majority of transaction are not introduced from reachable nodes, Biryukov et al. (2014) proposed an attack to discover the entry nodes of an unreachable node and relate transaction first observed from multiple entry nodes to the unreachable node (see Section 2.2.2 and Figure 2.2). This approach currently has a few weaknesses that can possibly limit the impact: the success of this attack depends on an unreachable node announcing its public IP address the network via connected entry nodes. This mechanism, however, is only useful for nodes that also accept incoming connections. Simply disabling this announcement for unreachable nodes would stop the attack, without further consequences for the reachability of the network. Disabling this announcement is already possible in recent versions of Bitcoin Core by unchecking the 'Accept incoming connections' option in settings.

Another weakness is that Biryukov et al. (2014) requires to establish 50 connections to all reachable nodes in order to achieve deanonymization of 11% of all transactions, however, doing so would damage the overall reachability of the Bitcoin network and the attack would be easily observable. For ethical and practical reasons we did therefore not consider doing this. Instead, we performed some research regarding the viability of this approach with the limitation of a maximum of 4 connections to all Bitcoin servers. Unfortunately, the proposals described here were

[1] `https://www.reddit.com/r/btc/comments/42k7ka/all_about_thin_blocks_in_xt/`

not tested sufficiently during the limited time of this research, so it is left for further research.

1. Where Biryukov et al. (2014) managed to "catch" a sufficient number of 3 entry nodes or more that were used to introduce a transaction in 77% of all transactions, *Bitsensory* only caught sufficient entry nodes in 27% of all transactions during an experiment in which 100 transactions were created by an unreachable node (see also Figure A.1). In order to compensate for this, we propose to use observations from multiple transactions that were created during the same session. As shown in Table A.1, 5 transactions will suffice, but the number of entry nodes that can be reliably detected increases when more transactions are available.

2. To allow discovering the entry nodes of an unreachable Bitcoin node, we propose to reverse the process of discovering the entry nodes of an unreachable Bitcoin client: for a given set of entry nodes that belong to the same entity with a high probability we see which unreachable Bitcoin client has most likely connected to those set of nodes by analyzing address messages. Instead of limiting propagation of `ADDR` messages as done by Biryukov et al. (2014), we look at the timestamp of a received address message. If the message is older than a few seconds, we disregard it.

   The probability of receiving at least n correct addresses for a given entry set of 8 nodes:

   $$\text{binompdf}(m, 8, n)$$

   where m is the probability of receiving the relevant `ADDR` message instead of another connected node:

   $$m = 2 \cdot \frac{\text{number of connections from attacker}}{\text{total number of connections}}$$

   As an alternative to detecting the entry nodes of unreachable Bitcoin nodes, it would also be possible to conduct a sybil attack to increase the probability that an unreachable node connects to an attacker's node.

	Number of transactions						
	1	2	5	10	20	50	100
First 1 observation	0.75	1.54	**3.14**	**5.16**	**6.23**	**7.44**	**8**
First 2 observation	1.1	2.22	**4.34**	**4.8**	**6.61**	**7.84**	**8**
First 3 observation	1.35	2.53	**3.87**	**4.55**	**6.78**	**7.89**	**8**
First 5 observation	1.51	2.83	**3.02**	**4.84**	**6.97**	**7.82**	**8**
First 10 observation	1.62	1.47	2.69	**5.14**	**6.68**	**7.89**	**8**

Table A.1: Number of transaction needed to discover sufficient entry nodes (averages)