Tracing Bitcoins Across Privacy Enhancing Technologies

Master-Thesis von Andreas Rothenhäuser aus Miltenberg
Tag der Einreichung:

1. Gutachten: Prof. Dr. Stefan Katzenbeisser
2. Gutachten: Spyros Boukoros, Nikolaos Alexopoulos

TECHNISCHE
UNIVERSITÄT
DARMSTADT

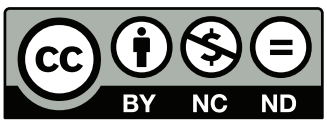
Computer Science
Security Engineering Group

Tracing Bitcoins Across Privacy Enhancing Technologies

Vorgelegte Master-Thesis von Andreas Rothenhäuser aus Miltenberg

1. Gutachten: Prof. Dr. Stefan Katzenbeisser
2. Gutachten: Spyros Boukoros, Nikolaos Alexopoulos

Tag der Einreichung:



Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 26.11.2018

(Andreas Rothenhäuser)

Abstract

Test

Contents

List of Figures

List of Tables

1 Introduction

The importance of looking at money flows in criminal prosecution was already known back in 80 BC, when Marcus Tullius Cicero phrased the question "Cui bono?" - "who benefits?" in his speech "Pro Roscio Amerino" [1]. Since then this idea has evolved to the wide field of financial investigation. Today following the money trail to identify suspects, find backers or explore the hierarchy of a criminal organization is considered a basic task by law enforcement authorities around the world. The ongoing Trump-Russia investigation of Robert Mueller is just one example of how the money trail can put pressure on criminals of all kinds. Paul Manafort and Michael Cohen are just two of the names on a long list of people already convicted for tax fraud, bank fraud, conspiracy, identity fraud, etc., because of Mueller's (financial) investigations [1, 2].
In 1994 the Internal Revenue Service (IRS) published a whole book dedicated to the art of financial investigation. The book was written as a guide for investigators, summarizing techniques that can be applied to almost any criminal case. Not only does it showcase how certain financial information can be interpreted and leveraged to build a successful case against a suspect, but it also lists potential sources of such information. Financial institutions usually apply Know Your Customer (KYC) policies in order to meet the Anti-Money Laundering (AML)/Combating the Financing of Terrorism (CFT) regulations of their home country or the countries they do business in. That means they collect personal information, like names, addresses or solvency about every customer and business partner they engage with. Furthermore they keep track of the financial transactions they do on behalf of their customers in bank statements. This is why the authors of the book note that financial institutions "are probably the single most important source of information available to the financial investigator" [3].
The recent rise of the so called cryptocurrencies like Bitcoin poses a major problem to law enforcement. In 2012 the Federal Bureau of Investigation (FBI) assesses, that "law enforcement faces difficulties detecting suspicious activity, identifying users, and obtaining transaction records" [4]. Financial institutions like the European Central Bank (ECB) follow along by stating problems with the enforcement of AML/CFT regulations [5]. Unlike traditional financial systems the Bitcoin protocol is designed to work in a decentralized and anonymous manner without any kind of financial institution to manage it. With the absence of a centralized institution collecting all the valuable information, investigators can not just walk in somewhere with a search warrant and retrieve information as needed. Thus many of the traditional techniques of financial investigation as described in the aforementioned book can not be applied in the world of Bitcoin & co. The greatest drawback from a financial investigator's point of view is the ability of criminals to hide behind Bitcoin addresses. Although it is not possible to hide transactions in the Bitcoin system, as long as no one knows which Bitcoin addresses belong to whom the transactions can not be linked to anyone in the real world.

[1] http://www.thelatinlibrary.com/cicero/sex.rosc.shtml

1.1 Contributions

The scheme I propose in this thesis strives to fill this gab, by solving the problem of mapping Bitcoin addresses to real world entities. Once the Bitcoin addresses of individual users are successfully identified the financial links between them can be trivially looked up on the Bitcoin blockchain by traversing the transaction graph. I will describe a solution tailored to the needs of law enforcement agencies, enabling them to

A) extract personally identifying information from the Bitcoin blockchain

B) cluster Bitcoin addresses according to their true owners based on the extracted information

The context of criminal prosecution induces some additional constraints, which have not received sufficient attention in previous work:

a) Legitimate approaches are limited to passive analysis of publicly available data, because active approaches can be problematic or even illegal in some legislation.

b) Any proposed scheme needs to make targeted deanonymization possible, because financial investigation is usually about incriminating or exonerating individuals.

c) The solution needs to be applicable in retrospect, because investigations can only start after a crime has been committed.

d) Finally the existence of privacy enhancing technologies like Bitcoin mixes should be accounted for, because the target group makes above-average use of them.

1.2 Outline

I will start by discussing previous work, that has been done in the field of blockchain analysis and the assessment of anonymity properties in the Bitcoin ecosystem in chapter 2. In chapter 3 I will detail some background aspects like the workings of blockchains and the Bitcoin protocol and describe the basic idea of stylometry. Chapter **??** will be about the precise problem statement of this thesis. Furthermore, I will explain the methodology of my approach. In chapter 5 I will describe the experiments I set up to assess the performance of my solution and provide the evaluation results. In the end I will give some ideas for future work on the topic and conclude the thesis in chapter **??** and **??** respectively.

2 Related Work

A lot of work has already been done regarding the anonymity properties of Bitcoins. Most of the papers that have been written on this topic can be roughly divided into three different categories based on their perspective and approach. The first category engages in the so-called blockchain analysis. As the name implies, blockchain analysis tries to take advantage of the information contained in the Bitcoin blockchain itself. In contrast to that, the second category applies techniques of network analysis. These approaches draw information from the analysis of the Bitcoin Peer-to-Peer (P2P) network traffic. The last category of papers deals with a variety of privacy enhancing technologies that have been proposed to improve the Bitcoin protocol over the years.

2.1 Blockchain Analysis

A very prominent approach based on blockchain analysis was already mentioned by the inventor of Bitcoin [6] and later used by Reid and Harrigan [7]. It utilizes the fact that Bitcoins, similar to real coins, need to be combined with other coins, if the amount that needs to be payed exceeds the value of any single coin the payee possesses. This combination of Bitcoins is done by creating so called multi-input transactions which have two or more coins as an input and at least one output.

> Example*: Alice owns three Bitcoin addresses* $A_1 = 5$*,* $A_2 = 2$*,* $A_3 = 3$*. She wants to send the amount of 10 Bitcoins to Bobs Bitcoin address B.*
> *Option 1: Alice issues three transactions* T_1*,* T_2 *and* T_3 *with one input and one output each.*
>
> T_1*:* $A_1 \Rightarrow B$
>
> T_2*:* $A_2 \Rightarrow B$
>
> T_3*:* $A_3 \Rightarrow B$
>
> *Option 2: Alice issues one multi-input transaction* T_1 *consuming all of her addresses' values at a time.*
>
> T_1*:* A_1
> $A_2 \Rightarrow B$
> A_3

The authors of [7] as well as others [8, 9] apply Bitcoin address clustering based on this observation in order to reduce the anonymity set of the Bitcoin system. The authors conclude, that multi-input transactions testify a common owner of all inputs of the transaction and thus map the corresponding Bitcoin addresses to the same superordinate entity. If an attacker now manages to identify one single Bitcoin address, the other addresses previously mapped to the same entity are compromised as well.
Another approach building on top of the aforementioned multi-input heuristic is to identify so called change addresses. This heuristic again relies on an inherent property of the Bitcoin protocol. Since unspent Bitcoins need to be spent in total or not at all, a common use case is to

add a special output to the transaction that receives the change (the money you get back, when you did not pay the exact amount). If the change address of a transaction is known, it can be mapped to the same entity as the inputs, reducing the anonymity set of the whole system even further. The change address heuristic was first explored in [9]. Later on it was refined in [8] to account for changes in the usage of Bitcoins.
Although both heuristics might have yielded good results in the early days of Bitcoins they are not suited very well to the problem under study. The multi-input heuristic is based on the assumption, that those transactions can only be issued by one single entity, whereas the change-address heursitic assumes, that each transaction necessarily has one change address. Back then these assumptions might have been reasonable, but the usage of Bitcoins has gone a long way since. Big mining pools regularly issue transactions with a lot of outputs to reward their miners. None of these outputs can be considered a change address, thus breaking the assumption of the change-address heuristic. In addition there are services that implement CoinJoin. The idea is basically to merge transactions of different users into a single, shared one in order to improve privacy (for more information see chapter 2). The resulting transactions have inputs that belong to different users, thus breaking the assumption of the multi-input heuristic. Both approaches exploit a common practice and still hold true in many cases today, but the number of cases to prove them wrong is growing quickly, making it increasingly difficult to satisfy constraint b) (see section 1.1).

2.2 Network Analysis

The two following approaches exploit the fact, that the Bitcoin P2P network is by its very nature open to everyone and is not designed to stop modified clients from eavesdropping on the network traffic.
In [10] the authors describe an approach that reveals the IP address of the original sender of a Bitcoin transaction by looking at suspicious relay patterns. They argue, that a well behaving client will always retransmit new Bitcoin transactions upon receipt, if the transactions are valid. Thus, if one peer on the network is the only one to relay a certain transaction, the sender has to be the creator of the transaction and all the corresponding Bitcoin addresses can be linked to the senders IP address.

> Example*: Suppose an attacker A is connected to every other active client on the Bitcoin network B and C. B and C are interconnected as well.*
> *T_1 is a normal transaction sent to all peers by B. It will be relayed by C and thus arrive at A via the blue as well as the red path. A can not determine who was the creator of T_1.*
> *T_2 is an invalid transaction sent by B. It will thus not be relayed by C and arrive at A only via the blue path. A can now conclude B was the true creator of T_2.*

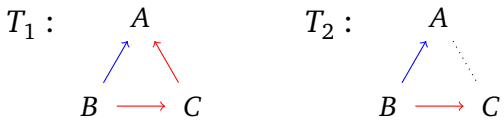


Figure 2.1.: Detection of lone relayer

Additionally, if many peers retransmit a transaction, but one peer is the only one to relay it multiple times the same assertion holds. In order to be able to observe such relay patterns across

the entire network an attacker ideally needs to have a connection to every participating client. This in turn requires modified client software as well as considerable storage and computing capacities.
A very similar approach was proposed by Biryukov in [11]. Again the attacker needs to connect to as much peers as possible in order to succeed. Conforming the protocol a newly connected peer will advertize its own IP address to eight random peers it directly connects to. These in turn will relay the new IP address to their own peers and so on. A completely connected attacker can now log the first eight peers he receives a new IP address from, having a good chance of retrieving the eight random peers of the newly connected peer. Using this eight peers as a fingerprint for following transactions received over the network can unmask the true originator of the transaction, because everything the attacker receives over this distinct eight peers is likely to originate from the same client. Again a link between the transaction and an distinct IP address is established.

> Example*: Suppose an attacker A is connected to every other active client on the Bitcoin network* 1*,* 2*,* 3*,* 4*. Now a new client N connects to two random peers* 1 *and* 3 *and advertizes its IP address.* 1 *and* 3 *will relay the IP and A will receive it via both. (See blue path.)*
> *If N issues a transaction* T_1 *afterward, A will receive it via the same two clients he received the IP address before. Since only N chose* 1 *and* 3 *as its initial peers, A can now conclude N being the original creator of* T_1*. (See red path.)*

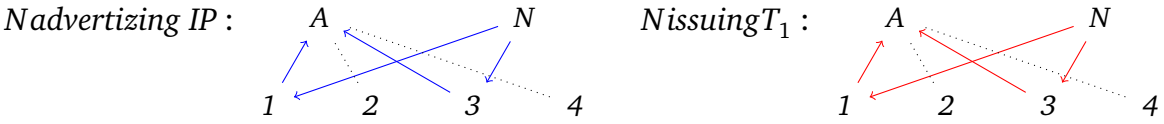


Figure 2.2.: Fingerprinting based on chosen peers

Both approaches can immediately deanonymize the originator of the transaction advertized to the network, but they have some serious drawbacks, which make them inapplicable to my scenario. First, they violate constraint a), because they require for large scale eavesdropping on network traffic. Secondly, the links between transactions and the originators IP address are uncovered at the moment the transaction is issued, which breaks constraint c).

2.3 Privacy Enhancing Technologies

The last category of papers does not deal with the Bitcoin protocol itself, but with privacy enhancing technologies that can be found in the Bitcoin ecosystem. Since they are meant to improve the anonymity of Bitcoin users, any work on assessing or attacking these also adds to the discussion on anonymity in the Bitcoin system. There are two important kinds of privacy enhancing technologies. The first is Bitcoin mixing and the second is the so called CoinJoin protocol.
The idea of Bitcoin mixing is based on the work on anonymous electronic messaging of Chaum [12]. A user *A* can send a certain amount of Bitcoin to the mixing service and the service will return the same amount to a new Bitcoin address of the user's choice. When the service is returning the money it will not use the coins formerly sent by *A*, but coins of another user it engaged with previously. Since *A* receives randomly choosen coins of the service provider's

funds, only the mixing service itself can link *A*'s inputs and outputs afterward. The service provider will usually demand a small fee for this and guarantuee to delete any paper trail on the transactions.

> Example*: A is sending Bitcoins to the mixing service and receiving coins of another user in return. Only the mixing service knows of the actual relationship between A and A′. On the Bitcoin blockchain this looks like B paid C′, A paid B′ and C paid A′.*

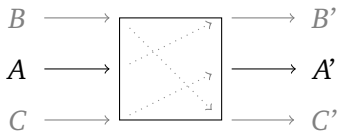


Figure 2.3.: Mixing service

The analysis of real world implementations of this scheme shows results that are twofold. On the one hand, it is indeed hard to correlate inputs and outputs of mixed coins, when Bitcoin mixing is properly implemented [13]. On the other hand many of the implementations out there do not provide proper unlinkability, due to a small user base or inproper handling of payouts [13, 14].
The second privacy enhancing technology, namely CoinJoin, was first proposed by Gregory Maxwell in 2013 [15]. It has found adoption by multiple projects later on[1]. CoinJoin requires multiple users to merge their transactions into one before advertizing it to the Bitcoin network. Every participant provides some inputs and specifies the desired outputs. All provided inputs and outputs are then combined in a single transaction, that every participant needs to sign. The resulting transaction is then fed to the Bitcoin network.

> Example*: Alice and Bob both want to pay anonymously and agree on the following transaction. They both provide two inputs totaling to 5 Bitcoins. By only looking at the transaction it is impossible to tell which outputs belong to Alice and which belong to Bob.*

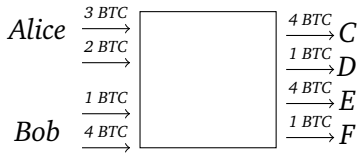


Figure 2.4.: CoinJoin transaction

Yanovich et al. have proven the problem of linking inputs to outputs in this scheme to be NP-hard [16].
Both examples show, that it is possible to achieve transaction unlinkability within the Bitcoin protocol. This makes it even harder for a financial investigator to track flows of Bitcoin. In contrast to the previously mentioned schemes, my approach is designed to cope with such privacy enhancing technologies without utilizing implementation specific weaknesses.

[1] `https://en.bitcoin.it/wiki/User:Gmaxwell/state_of_coinjoin`

3 Background

This chapter will explain some important background aspects of the thesis in more detail. In the first section I will describe the structure of the Bitcoin blockchain and how it is build by the community. Section two elaborates on the communication protocol Bitcoin relies on and highlights the characteristics of P2P networking. Section three introduces stylometry and discusses its different forms of application in computer science.

3.1 The Blockchain

In 2008 Satoshi Nakamoto introduced the now famous payment scheme 'Bitcoin' and with it the notion of a blockchain. The blockchain is actually what its name implies: A chain of blocks, which in turn are collections of transactions. This is why I will start explaining the blockchain by giving details about transactions and blocks first.

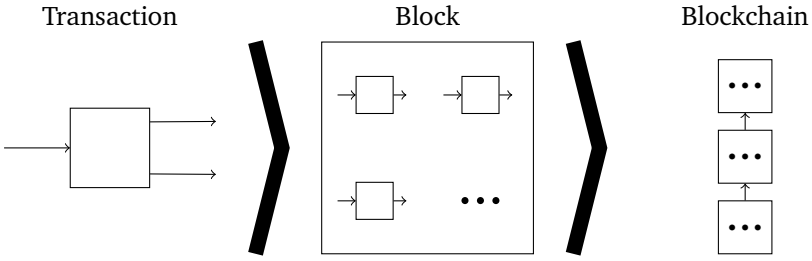


Figure 3.1.: Structure of the Blockchain

3.1.1 Structure of Transactions

Transactions are basically just statements like "Alice sends 10 Bitcoins to Bob". In order to provide the users anonymity, the clear names (like Alice and Bob) are actually replaced by cryptographic keys. Any user can create an infinite amount of such keys. To increase human readability Bitcoin keys are usually displayed in a special format called the Bitcoin address. For the sake of simplicity both terms are used interchangeably throughout this document. Technically speaking the amount of 10 Bitcoins is either the sum of all transaction inputs or all transaction outputs. A transaction contains a number of inputs and a number of outputs. The inputs each reference the distinct output of a previous transaction that will be consumed by the input. Furthermore they contain a script that will be called to verify the legitimacy of the "withdrawal" of the referenced transaction output. Each output contains the amount being spent and a script controlling who will be able to claim the amount. In most cases this script directly states the Bitcoin address of the receiver. I will call such an addresse the target addresses of this output.

3.1.2 Construction of Blocks

The transactions are collected by so-called miners. Miners verify all transactions, by executing the scripts and after that integrate them into a new block. By attaching a cryptographic hash of the previous block to the newly created one, they build a chain out of the blocks, the blockchain. Miners are allowed to add a so-called coinbase transaction to each block they mine, which will grant them some newly generated Bitcoins as reward for their work, but they have to compete with other miners as only the first valid block will be added on top of the blockchain. Calculating a valid block entails more than just gathering enough transactions and adding the hash of the previous block. In order to discourage malicious entities from adding nonsensical blocks to the blockchain a miner has to provide proof, that he spent considerable work on the creation of the new block. He does so, by adding a nonce to each block and changing its value until the resulting hash value of the block contains a certain amount of leading zeroes. The number of zeroes required is called difficulty. The difficulty is adjusted automatically to compensate for changing engagement of miners and ensure a constant growth of the blockchain. For a block to be accepted by the community it has to meet certain criteria:

1. Block can only contain transactions with verified inputs
2. Block must include the hash of the last accepted block
3. Block time needs to be "greater than the median timestamp of previous 11 blocks, and less than the network-adjusted time +2 hours."[1]
4. Block hash needs to have a number of leading zeroes according to the current difficulty

3.1.3 Characteristics of the Blockchain

The structure of the blockchain guarantees for some important properties like inmutability, publicly verifyable transactions and prevention of double spending.
Inmutability is ensured by the combination of two facts. First, changing the contents of a previously accepted block entails redoing the work of finding the correct nonce. Else, the community would not accept the manipulated block as valid. Secondly, since the hash of the original block has been integrated into the following block, the entire blockchain has to be recomputed from the point of the manipulation on. Otherwise all following blocks would be cut off the chain, which would immediately invalidate all included transactions. Honest miners will always advance on the longest chain they find. Hence the cheater will have to outpace all the other miners alone, if he wishes the community to accept his recomputed version of the blockchain. While manipulation would be feasible in theory, as long as the majority of computational power is on the side of honest players, the chances of this to work out are very small.
In addition, transactions are publicly verifyable, because the blockchain itself is open to the public and contains every transaction the community ever agreed on. Everyone has the option to look up and verify any transaction.
Double spending is prevented by requiring every transaction statement to explicitly name the past transactions that source the amount to be spent in the newly advertized transaction. For example, if Alice wants to send 10 Bitcoins to Bob, Alice needs to name previous transactions,

[1] https://en.bitcoin.it/wiki/Block_timestamp

that in turn sent the amount of 10 Bitcoins to her. This way she can prove, that she indeed owns the 10 Bitcoins she wants to send. It does not matter which and how many transactions Alice names, as long as she has not called upon the same resources before and they total to at least the amount of 10 Bitcoins. Again all this constraints can be validated by anyone, by looking up all related transactions in the blockchain.

3.2 The Bitcoin Network

Bitcoin clients connect to each other via the Bitcoin Peer-to-Peer (P2P) network. Peer-to-peer means, that all messages are exchanged by the clients directly. There is no central entity organizing traffic flow and decisions need to be made by consensus of the community. The Bitcoin network relies on a gossip protocol to spread information.

3.2.1 Connecting to the Network

At the first contact each node connects to a number of entry peers chosen randomly from a list of known peers. Client software usually implements hardcoded fallback lists to call upon, if the list of known peers is empty or outdated. Whenever a node establishes a new connection to a peer, it advertizes its own IP address to the peer. The peer will store the IP address in its list of known peers and eventually relay it to its own entry peers and so on. This way the IP address of each newly connected node will spread across the whole network.

3.2.2 Downloading the Blockchain

Every Bitcoin client maintains a local copy of the Bitcoin blockchain. If this copy is outdated or has not been created yet, the client will ask its directly connected peers for updates. Given that the peers can provide newer data, they will answer the request with the required data. Before the client integrates this data into its local blockchain, it will check every block and the contained transactions for validity.

3.2.3 Issuing Transactions

To make Bitcoin payments, the client needs to create a valid transaction and send it to its directly connected peers. Similar to IP address propagation, the peers receiving a new transactions will relay them to their respective peers. To reduce the network load, transactions will only be relayed, if they pass certain validity checks.
Miners are alway listening for such new transactions and collect them for integration in a new block. One can chose to draw on a tiny bit of more resources in a transaction input, than is actually required for the output. The difference will automatically be regarded as a fee to reward the miner, who adds the transaction to a block. Usually a larger fee results in faster integration of the transaction into the blockchain.
Miners will advertize new blocks to the network in the same way users do with transactions. If the block satisfies all requirements, each client will add it to its local copy of the blockchain upon receipt. The transaction can now be considered confirmed. In some rare cases two miners

will succeed in finding a new block at almost the same time. When this happens, the miners may freely choose the block they want to build their next block on. As different miners may choose differently, this can lead to diverging branches of the blockchain. The branch supported by the majority of the miners will eventually outpace the other and the shorter branch will be abandoned. This is also the reason, why practioneers usually wait a few blocks before they consider a certain transaction confirmed by the community irreversibly.

3.3 Stylometry

Stylometry builds on pattern recognition to identify the author of a work. It has successfully been applied to various fields, like writing, music, painting and computer code.
Unique patterns in style allow an analyst to differenciate between single authors. Depicting the style by numerical or logical features enables computers to aid in analysing it. A typical stylometric analysis would include the phases of pre-processing, feature crafting, feature extraction and decision making.

3.3.1 Pre-Processing

In the pre-processing phase the data analysed needs to be transformed to a computer-readable shape. Depending on the application field this can require multiple steps, where in the end the data needs to be in a digitalized and well defined format.

3.3.2 Feature Crafting

The phase of feature crafting is tightly coupled with the data itself. The goal of this phase is to identify reasonable and discriminative features to describe the style of a single piece of data. For example, if you want to identify the author of a business letter the valediction might not be as discriminating a feature as the spelling mistakes, because there are only very few valedictions commonly used, but many mistakes one can make. Adding to the quality of spelling mistakes as example feature is the fact, that not knowing how to spell out certain words is a very personal issue. On the other hand looking at spelling mistakes, if you want to find the painter of some picture might not be a good idea as well, because chances are good there is no text to find mistakes in at all. The total of all features identified in this phase is called the feature set.

3.3.3 Feature Extraction

Once the analyst has decided on a feature set, the phase of feature extraction can take place. In this phase every piece of data is looked at. For each feature of the feature set a (usually) numerical value is extracted and added to the feature vector. The feature vector is subsequently used to describe the style of the corresponding piece of data.

3.3.4 Decision Making

In the last phase a decision regarding the author of each piece of data needs to be made. The outcome of this phase can slightly differ based on the concrete problem statement. For author-

ship verification a simple 'yes' (the hypothesis of *A* being the author is accepted) or 'no' (the hypothesis of *A* being the author is rejected) can be enough. For authorship attribution one of many authors needs to be assigned to every piece of data. Both problems require a reference work for each author to compare the computed feature vectors to. In a more open scenario it might even be enough to just cluster the single pieces of data by the distance to each other to discover the number of different authors.
It is important to note that, no matter what the concrete question is, the answer can only be an estimate based on the conformity of the extracted feature vector with the reference vector.

Deep Learning and Interpretability

It is increasingly popular to combine stylometry with unsupervised machine learning techniques to automatically find features and make decisions. So-called deep learning algorithms based on neuronal networks are considered the state-of-the-art approach at the time of this writing. The usage of such techniques heavily lightens the workload of analysts, because little to no human intervention is required in the feature crafting phase. The only thing needed is lots of data to train the system.
This goes at the expense of interpretability, since sometimes not even experts can explain why certain features have been chosen and why they add to the final decision [17, 18]. Currently there is no solution to this problem, but a lot of efford has been directed at finding ways of establishing trust in black box models [18, 19, 20, 21]. Depending on the application the lack of an explanation may or may not be an acceptable property.

4 Uncovering Bitcoin Owners

4.1 Problem Statement

Before explaining the exact problem under study, I first want to draw a picture of a financial investigator's situation when being confronted with a case involving Bitcoin. As with normal cases the question of who paid whom can be rather interesting for attorneys as well as judges. Thus the investigator is frequently confronted with it and sometimes all he has is a single Bitcoin address or suspicious Bitcoin transaction from a seized email conversation to start with. With a bit of luck he might even have access to an unlocked Bitcoin wallet containing multiple Bitcoin addresses of a suspect. But despite of every Bitcoin transaction being public, the answer is not necessarily easy to find, because transaction inputs and outputs are linked to Bitcoin addresses, not people. Especially in the criminal world it is common practice to strictly separate assets by categories. The wallet for "clean money" to pay friends and to buy legal goods might be installed without any protection on the smartphone to have 24/7 access to it. But the wallet for "dirty money", the one for paying the local drug dealer and selling stolen goods, resides password protected on a fully encrypted hard drive. Money transfers between the two are obscured by convoluted paths and sometimes the use of mixing services and the like. The real challenge for the investigator is not to follow the path of the Bitcoins through the blockchain. This could easily be accomblished by blockchain explorers[1]. The real challenge is to find all the Bitcoin addresses belonging to a suspect in the first place. For this reason, the primary goal of this thesis is to design a scheme, which enables a financial investigator to identify address clusters belonging to the same real world entity.

4.2 Methodology

In sections 2.1 and 2.2 I have discussed some prominent approaches to map Bitcoin addresses to real world entities, but in one or the other way they all fail to provide assistance in this specific task. Some are based on assumptions that do not hold true in the context of criminal prosecution, where mixing services are standard and wallet software with integrated CoinJoin functionality might be used. Others require authority and ressources, that go far beyond those of a simple investigator. The approach I will describe in the following tries to overcome this limitations by entirely relying on passive blockchain analysis. It can be applied even in the presence of privacy enhancing technologies that break the direct link on the blockchain between the true sender and the true receiver of some Bitcoins. This is achieved by profiling usage patterns of Bitcoin addresses rather than looking at their positioning within the blockchain. I assume, that there is at least one known Bitcoin address or transaction to start from. My scheme enables a financial investigator to find other Bitcoin addresses of the same person and cluster the Bitcoin addresses of the relevant section of the blockchain in order to uncover hidden links between the real world entities behind the clusters. As I generally follow a stylometric approach

[1] https://www.blockchain.com/explorer
https://blockexplorer.com

to detect the true owners of Bitcoin addresses, the analysis roughly follows the four phases mentioned in section 3.3.

4.2.1 Data Aquisition and Pre-Processing

As I am implementing a blockchain analysis scheme, the only source of information will be the Bitcoin blockchain itself. As mentioned earlier every Bitcoin client already has the capability of connecting to the Bitcoin network and downloading a local copy of the blockchain. The original Bitcoin client stores this data sequentially in a binay format called the block-files. For a couple of reasons I decided against reusing this local copy.

1. Freshness of the data can only be assured by keeping the client running and connected to the network

2. A sequential data structure is unsuitable for fast lookups

3. A dependency on the Bitcoin client introduces an undesirable complexity for the user

Given the fact, that fast lookups and easy traversal of Bitcoin transactions are crucial to design an efficient tool, I decided to store the data in a graph database. In order to keep dependancies on other tools small, network communication is realized via a wrapper library around the Bitcoin Application Programming Interface (API).

Data Model

Transforming the blockchain data to a graph that can be stored in a graph database is straight forward. All transactions contained in the blockchain are linked by their inputs and outputs. By connecting each transaction input with the output of the previous transaction stated therein an acyclic graph is constructed, which can be fitted into the database directly. The graph contains the transaction bodies as vertices and the inputs and outputs as edges. Table 4.1 lists the details of the database objects and their fields. I refer to the resulting graph as the Bitcoin transaction graph throughout the thesis.

Name	**Description**	**Type**
Transaction (Vertex)		
Hash	A hash to uniquely identify the transaction	string
BlockTime	The block time of the corresponding block	DateTime
Coinbase	A flag set for coinbase transactions	bool
Link (Edge)		
amount	The value transferred over this link	long
tAddr	The target address receiving the amount	string
sN	The index of the referenced transaction output	long

Table 4.1.: Database scheme

4.2.2 Feature Crafting

Since all people are creatures of habit, they unwittingly imprint a recognizable pattern on everything they do. These patterns can be unveiled, when closely looking at the right features of their doings. In the following I will present the features I use to profile Bitcoin addresses and their users. All features refer to Bitcoin addresses and will be extracted from every block and transaction the respective address occurs in. A quick overview of the entire feature set can be found in table 4.2.

Feature	**Description**	**Distance**
Time-Based Features		
F_{DoW}	Vector with one flag per day of the week	$D = \frac{\sum(x_i \oplus y_i)}{\sum x_i + \sum y_i}$
F_{HoD}	Vector with one flag per hour of the day	$D = \frac{\sum(x_i \oplus y_i)}{\sum x_i + \sum y_i}$
F_{CT}	Vector with one counter per hour of the day	$D = \frac{\sqrt{\sum(x_i - y_i)^2}}{\sum x_i + \sum y_i}$
Amount-Based Features		
F_A	Scalar expressing the average amount received	$D = \frac{\lvert x-y \rvert}{x+y}$
F_{TA}	Scalar expressing the average input of transactions	$D = \frac{\lvert x-y \rvert}{x+y}$
Features Based on Social Network		
F_{H1}	List of addresses cooccuring as transaction inputs	$D = \begin{cases} 0 & X = Y \\ 1 & otherwise \end{cases}$
F_{H2}	List of identified change addresses	$D = \begin{cases} 0 & X = Y \\ 1 & otherwise \end{cases}$
F_{SN}	List of addresses cooccurring within transactions	$D = \frac{\lvert X \cap Y \rvert}{\lvert X \cup Y \rvert}$
F_{TS}	Vector with two elements: one being the average number of inputs, one the average number of outputs.	$D = \frac{\sqrt{\sum(x_i - y_i)^2}}{\sum x_i + \sum y_i}$

Table 4.2.: Feature set

Time-Based Features

Although the block time is not very accurate compared to other time measures it can still be used to construct meaningful features. On the one hand the whole analysis can be considered as a closed system which is completely independant of real-time scenarios. On the other hand depicting order and regularity do not require precise timestamps.
People usually have a more or less fixed daily routine mainly imposed by individual working hours and other duties. The combination of the first two features aims to describe the resulting

schedule a Bitcoin user follows when interacting with the Bitcoin network. The DayOfWeek feature (F_{DoW}) extracts the day of the week and the HourOfDay (F_{HoD}) feature extracts the hour of the day from the block time. Both features are represented by vectors of flags indicating at which day and which hour the address occured on respectively.

> Example*: Suppose Alice usually does her Bitcoin business on saturday and sunday in the evening. The corresponding vectors extracted from her Bitcoin addresses might look similar to this:*
>
> $F_{DoW} = (0;0;0;0;0;1;1)$
> $F_{HoD} = (0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;0;1;1;1;0;0;0)$

The CommonTimes feature (F_{CT}) in addition adds weights according to the number of occurences of a Bitcoin address within a certain hour. It is designed to depict the intensity of Bitcoin usage at a certain time. Opposing F_{DoW} and F_{TS} the result is a vector of scalars, not flags, simply counting up per occurence.

> Example*: Suppose Alice used a Bitcoin address ten times. Seven times in the evening, two times in the morning and once at noon. The corresponding vector might look as follows:*
>
> $F_{CT} = (0;0;0;0;0;0;0;1;1;0;0;0;1;0;0;0;0;0;0;2;4;1;0;0;0)$

Amount-Based Features

Just like in real life, there are Bitcoin users with more and ones with fewer money at their disposal. There are investors moving huge amounts of Bitcoin from A to B as well as nerds that just want to test a new interesting technology for fun with just a few dollars. The features of this category try to differentiate between users based on the amounts received.
The Amounts feature (F_A) extracts the average amount of Bitcoins transfered to a certain Bitcoin address.

> Example*: Suppose there exist three transactions T_1, T_2, T_3 sending Bitcoins to a certain Bitcoin address. The transaction outputs targeted towards the Bitcoin address amount to $A_{T_1} = 2$, $A_{T_2} = 5$, $A_{T_3} = 6$. This is what the resulting feature would look like:*
>
> $F_A = \frac{2+5+6}{3} = 4,\overline{3}$

The TotalAmounts feature (F_{TA}) looks at the whole transaction, not only a certain output, to get an idea of the total amounts being transfered by the corresponding Bitcoin user. It is the average amount of the sum of all inputs to a transaction the respective Bitcoin address occurs as input.

> Example*: Suppose a Bitcoin address is used two times along with other addresses to source huge transactions. The inputs of the first transaction total to 100 Bitcoins and the inputs of the second transaction total to 110 Bitcoins. The resulting feature would be:*
>
> $F_{TA} = \frac{100+110}{2} = 105$

Since Bitcoins can be split up to 1/100000000 there is a very high number of possible outcomes for the amount-based features. Furthermore considering an address that receives 0.00000001 Bitcoins different from one receiving 0.00000002 Bitcoins does not support the idea of identifying Bitcoin users by their average earnings or spendings very well. For this reason only two significant figures of each extracted amount will be considered during the calculation of F_A and F_{TA}.

Features Based on Social Network

I consider all Bitcoin addresses cooccurring with the target Bitcoin address as its social network. It has been shown before, that social networks tend to evolve small communities of strongly interconnected nodes [22, 23]. The following features leverage this fact by analysing the nearness of Bitcoin addresses regarding the structure of and affiliation to their respective social networks. The first feature utilizes the idea of multi-input clustering as described in section 2.1 to compile a list of cooccuring Bitcoin addresses. The cooccurences are first compiled by transactions and collapsed to independent clusters afterward. Following the wording of [8, 9] I call this feature Heuristic1 (F_{H1}).

> Example*: Suppose address* $1AbC$ *occurrs within two multi-input transactions* T_1 *and* T_2*, the resulting feature would be:*
>
> $$T_1 : \begin{matrix} 1AbC \\ 1DeF \end{matrix} \Rightarrow 1UvW \qquad T_2 : \begin{matrix} 1AbC \\ 1GhI \end{matrix} \Rightarrow 1XyZ$$
>
> $$F_{H1} = \{'1AbC', '1DeF', '1GhI'\}$$

Accordingly the Heuristic2 feature (F_{H2}) utilizes the idea of change address identification to again compile a list of related Bitcoin addresses. The input addresses are first grouped together with their respective change address and afterwards collapsed to independant clusters.

> Example*: Suppose there are two Bitcoin transactions* T_1 *and* T_2*. The first output of each transaction has been successfully identified as the change address. The resulting feature would look like this:*
>
> $$T_1 : \begin{matrix} 1AbC \\ 1Rn1 \end{matrix} \Rightarrow 1Ch1 \qquad T_2 : \begin{matrix} 1AbC \\ 1Rn2 \end{matrix} \Rightarrow 1Ch2$$
>
> $$F_{H2} = \{'1AbC', '1Ch1', '1Ch2'\}$$

F_{H1} and F_{H1} try to express nearness of Bitcoin addresses by identifying an alleged common owner of the addresses, which is a rather high requirement. The SocialNetwork feature (F_{SN}) assumes, that a certain nearness is allready implyed by cooccurences of any kind within transactions. Thus, F_{SN} extracts just any Bitcoin address that ever occurred with a certain address of interest.

> Example*: Suppose there are two Bitcoin transactions* T_1 *and* T_2 *on the blockchain, which contain the Bitcoin address of interest* $1AbC$*. The SocialNetwork feature would look as follows:*

$$T_1 : 1AbC \Rightarrow \begin{matrix} 1DeF \\ 1GhI \end{matrix} \qquad T_2 : 1AbC \Rightarrow \begin{matrix} 1MnO \\ 1JkL \end{matrix}$$

$$F_{SN} = \{'1AbC','1DeF','1GhI','1JkL','1MnO'\}$$

Occasional Bitcoin users tend to issue small transactions with sometimes no more than one input and one output. More professional Bitcoin users, like exchanges, mining pools or other service providers are more likely to optimize their transactions. A mining pool for example will pay all participating miners within a single transaction to avoid additional fees for many single transactions. The TransactionShape feature (F_{TS}) extracts the number of inputs and the number of outputs to depict this behaviour.

Example*: Given the example transaction T_1, the corresponding F_{TS} would look like:*

$$T_1 : \begin{matrix} 1AbC \\ 1DeF \end{matrix} \Rightarrow \begin{matrix} 1GhI \\ 1JkL \\ 1MnO \end{matrix}$$

$$F_{TS} = (2,3)$$

4.2.3 Feature Extraction

The goal of feature extraction is to collect all the information specified in section 4.2.2 from the actual data. By doing so an array of numerical and logical values is compiled, which represents the unique mark the owner of the corresponding Bitcoin address left on it. This array is referred to as the Bitcoin address's 'feature vector' or simply 'profile'. Since the data basis is the Bitcoin transaction graph, but all the features refer to addresses, some feature specific pre-processing is required before extraction is possible. Depending on the feature under scrutiny more or less data is derived from the graph database beforehand. Table 4.3 lists the entries that need to be looked up by feature. The alongside picture shows an edge containing the Bitcoin address of interest.
The following term definitions are supposed to increase the readability of the subsequent descriptions:

Relevant edge:
: An edge containing the Bitcoin address under scrutiny

Target Transaction:
: The vertex of the Bitcoin transaction graph a certain edge points to

Source Transaction:
: The vertex of the Bitcoin transaction graph a certain edge originates from

Outgoing Address:
: An address stored in an outgoing edge of a vertex

Incoming Address:
: An address stored in an incoming edge of a vertex

Feature	Resources
F_A	1
F_{Dow}	1, A
F_{HoD}	1, A
F_{CT}	1, A
F_{TA}	1, 2
F_{H1}	1, 2
F_{SN}	1, 2, 3, 4
F_{TS}	1, 2, 3, 4
F_{H2}	1, 2, 3, 4, +

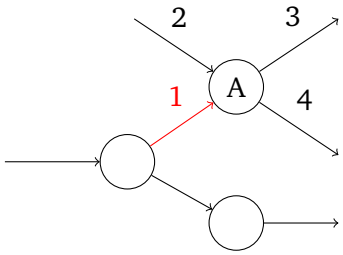


Table 4.3.: Required data resources for feature extraction

- For F_A it is sufficient to look up all relevant edges. The feature value is computed by simply averaging all transferred amounts specified in the relevant edges.
- The time-based features (F_{DoW}, F_{HoD}, F_{CT}) require to query all target transactions of relevant edges, because the block time is stored within vertices. The respective days of the week and hours of the day are extracted from the returned list of vertices. In the case of F_{DoW} and F_{HoD} a flag is set on the first occurence of every day or hour respectively. In the case of F_{CT} a conter is maintained for each hour of the day, which is increased on every occurence.
- Both F_{TA} and F_{H1} require the incoming edges from each target transaction of the relevant edges. Thus for each relevant edge a list of neigboring edges is retrieved from the database. In the case of F_{TA} the feature value is calculated as the average sum of the amounts of neighboring edges including the amount of the relevant edge itself. In the case of F_{H1} the feature value is a single, deduplicated list of all Bitcoin addresses contained in the retrieved lists.
- In addition to the neighboring incoming edges F_{SN} and F_{TS} also require the outgoing edges of each target transaction. While F_{TS} simply counts incoming and outgoing edges to derive the average count, F_{SN} extracts the contained addresses to compile a single list of all adjacent Bitcoin addresses.
- F_{H2} is a special case, because it does not only require the incoming and outgoing edges of each target transaction, but also needs the total count of occurences of every single outgoing address. The feature value is the list of all outgoing addresses which have been identified as change addresses.

Defining a Scope

In theory feature extraction could be applied to all addresses in use. Regarding the problem statement of this thesis, this is the worst case scenario, as the anonymity set is maximized. The evaluation presented in 5 tries to reflect this case, by relying on labeled data distributed over 8 years (ranging from 2010 to 2017). In an actual use case I strongly recommend to narrow the analysis down to a smaller section of the blockchain. If for example the time of a crime is known

or can be roughly estimated by looking at the block time of a suspicious transaction, one can decide to just analyse a time window of a few weeks before or after. This reduces the number of real world actors and thus increases the accuracy of the results. If an investigator is able to rule out the usage of privacy enhancing technologies beforehand or simply wants to analyse a straight path of transactions visible on the blockchain, he might even choose to only select Bitcoin addresses from the transaction graph to a certain depth, starting from a transaction of interest. Generally spoken, the further the down selection is narrowed, the smaller is the anonymity set of a suspect, making the task of correctly mapping Bitcoin addresses to their true owners ever more easy.

4.2.4 Decision Making

In the previous section I have described how to transform Bitcoin addresses and transactions to a feature vector, which carries a personal mark of the true owner of an Bitcoin address. In this section I will explain how these feature vectors can be used to answer the central question of this thesis on how Bitcoin addresses can be clustered with respect to their true owners. From a technical perspective, this question contains two aspects that need to be considered.

1. The task is to form groups of Bitcoin addresses that are close to each other
2. Closeness should be considered with regard to the true owner of each address

Distance Calculation

Since the priviously designed feature vectors represent characteristics of the address owners, they can be used to express closeness or to put it the other way around distance between address owners. The example below uses a reduced feature set $\vec{v}_{schema} = (F_{DoW}, F_A, F_{H1})$ to showcase how the distance is calculated. Calculation is done pair wise and on a per feature basis. The distance functions of all features yield results ranging from 0 to 1. The total distance between two feature vectors is defined as the average distance of all features. Table 4.2 contains the complete list of distance functions.

Example*: Distance calculation of $\vec{a}$ and $\vec{b}$:*
$\vec{a} = ((0,0,0,1,0,0,1),(2100000),(1AbC,1DeF))$
$\vec{b} = ((1,0,0,0,0,0,0),(170000000),(1GhI,1JkL,1MnO,1PqR))$

$$D_{DoW} = \frac{(0 \oplus 1) + (0 \oplus 0) + (0 \oplus 0) + (1 \oplus 0) + (0 \oplus 0) + (0 \oplus 0) + (1 \oplus 0)}{(0+0+0+1+0+0+1) + (1+0+0+0+0+0+0)} = 1 \quad (4.1)$$

$$D_A = \frac{|2100000 - 170000000|}{2100000 + 170000000} \approx 0.98 \quad (4.2)$$

$$D_{H1} := (1AbC, 1DeF) \neq (1GhI, 1JkL, 1MnO, 1PqR) \Rightarrow 1 \quad (4.3)$$

$$D_{Total} = \frac{1 + 0.98 + 1}{3} \approx 0.99 \quad (4.4)$$

Clustering

Having defined a suitable distance metric for Bitcoin addresses, the process of clustering is straightforward. Since the number of resulting clusters is not known beforehand, I decided to use hierarchical clustering. Hierarchical clustering can be done in two different ways.

1. The top-down approach starts with all data points being in one single cluster at the base layer of hierarchy. The algorithms iteratively splits the biggest cluster into two. A new cluster is formed by separating the data point with the highest distance to any other data point in the cluster being split. Afterward all data points that are closer to the new cluster, than to the old are shifted to the new cluster. The algorithm will repeat to split clusters until every cluster contains only one data point or some stop criterion is met.

2. The bottom-up approach proceedes in reverse. It starts with one cluster per data points at the base layer and iteratively merges the two clusters with the smallest distance inbetween. At the final layer the algorithm produces only one cluster containing all data points.

In the problem under study the number of clusters is expected to be rather high with respect to the number of data points. This is why the bottom-up approach, often referred to as Hierarchical Agglomerative Clustering (HAC) is used. The distance between each cluster, which needs to be calculated in each step of the algorithm, is defined as the average distance between every pair of data points.

> Example*: Clustering of three Bitcoin addresses* $A = 1AbC$*,* $B = 1DeF$ *and* $C = 1GhI$ *with known distances.*
>
> $D_{AB} = 0.1$*,* $D_{AC} = 0.2$*,* $D_{BC} = 0.3$
>
> Layer 1 $((A),(B),(C))$ *Distance: 0*
>
> Layer 2 $((A,B),(C))$ *Distance: 0.1*
>
> Layer 3 $((A,B,C))$ *Distance: 0.15*

The maximum distance for clusters to be merged increases in each layer. It can be considered as a dissimilarity metric of data points within the same clusters. By choosing a suitable distance metric as a stop criterion for the algorithm the output can be tuned to contain the desired results.

5 Experiments + Evaluation

Bibliography

[1] T. McCarthy, "Trump and 'collusion': what we know so far about mueller's russia investigation," 2018.

[2] J. Rubin, "Mueller follows the money trail," 2018.

[3] D. Vogel *et al.*, *Financial Investigations: A Financial Approach to Detecting & Resolving Crimes*. DIANE Publishing, 1994.

[4] FBI, "Bitcoin virtual currency: Intelligence unique features present distinct challenges for deterring illicit activity," 2012.

[5] ECB, "Virtual currency schemes – a further analysis," 2015.

[6] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[7] F. Reid and M. Harrigan, "An analysis of anonymity in the bitcoin system," in *Security and privacy in social networks*, pp. 197–223, Springer, 2013.

[8] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 127–140, ACM, 2013.

[9] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *International Conference on Financial Cryptography and Data Security*, pp. 34–51, Springer, 2013.

[10] P. Koshy, D. Koshy, and P. McDaniel, "An analysis of anonymity in bitcoin using p2p network traffic," in *International Conference on Financial Cryptography and Data Security*, pp. 469–485, Springer, 2014.

[11] A. Biryukov, D. Khovratovich, and I. Pustogarov, "Deanonymisation of clients in bitcoin p2p network," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 15–29, ACM, 2014.

[12] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 24, no. 2, pp. 84–90, 1981.

[13] M. Moser, R. Bohme, and D. Breuker, "An inquiry into money laundering tools in the bitcoin ecosystem," in *eCrime Researchers Summit (eCRS), 2013*, pp. 1–14, IEEE, 2013.

[14] T. de Balthasar and J. Hernandez-Castro, "An analysis of bitcoin laundry services," in *Nordic Conference on Secure IT Systems*, pp. 297–312, Springer, 2017.

[15] G. Maxwell, "Coinjoin: Bitcoin privacy for the real world," 2013.

[16] Y. Yanovich, P. Mischenko, and A. Ostrovskiy, "Shared send untangling in bitcoin," *The Bitfury Group white paper*, 2016.

[17] W. Knight, “The dark secret at the heart of ai,” 2017.

[18] D. Gunning, “Explainable artificial intelligence,” tech. rep., Technical Report DARPA-BAA-16-53, DARPA Broad Agency Announcement, 2016.

[19] Z. C. Lipton, “The mythos of model interpretability,” *arXiv preprint arXiv:1606.03490*, 2016.

[20] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.

[21] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec, “Interpretable & explorable approximations of black box models,” *arXiv preprint arXiv:1707.01154*, 2017.

[22] D. J. Watts and S. H. Strogatz, “Collective dynamics of ‘small-world’networks,” *nature*, vol. 393, no. 6684, p. 440, 1998.

[23] M. Girvan and M. E. Newman, “Community structure in social and biological networks,” *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

6 Glossary

AML Anti-Money Laundering

API Application Programming Interface

CFT Combating the Financing of Terrorism

ECB European Central Bank

HAC Hierarchical Agglomerative Clustering

IRS Internal Revenue Service

KYC Know Your Customer

FBI Federal Bureau of Investigation

P2P Peer-to-Peer

A Some Appendix