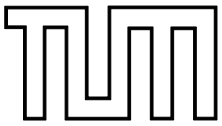
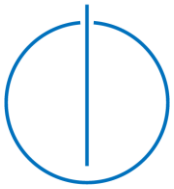


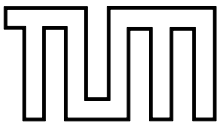
DEPARTMENT OF INFORMATICS
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Identification of Cross-Blockchain Transactions: A Feasibility Study

Patrick Nieves





DEPARTMENT OF INFORMATICS
TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis in Information Systems

Identification of Cross-Blockchain Transactions: A Feasibility Study

Identifizierung von Blockchain-übergreifenden Transaktionen: Eine Machbarkeitsstudie

Author:	Patrick Nieves
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Patrick Holl,
	Ulrich Gallersdörfer
Submission Date:	15. May 2018

I confirm that this master's thesis in information systems is my own work and I have documented all sources and material used.

Munich, 15. May 2018 _____

Signature

Abstract

Since the interest in blockchain technology has risen in recent years, the number of cryptocurrencies has increased. Due to this, there is an increasing need to exchange currencies. Several companies offer services that solve this problem in different ways. The transfers they execute are called cross-blockchain transactions. This kind of exchanges occasionally produce two single transactions, which are saved on the blockchains of the involved cryptocurrencies. As these ledgers are distributed over public peer-to-peer networks, the transaction data can easily be retrieved and analyzed. However, exchange information linking two transactions to each other is not publicly visible.

This thesis aims to identify cross-blockchain transactions, by matching single transactions saved on the different blockchains. First, we analyze in detail the processes taken by an exchange service when executing a trade. Based on this, we determine heuristics and design a recognition algorithm. We implement this algorithm within a tool and evaluate its output. Finally, we show to what extent the identification of cross-blockchain transactions is feasible.

Keywords:

Cryptocurrency, Blockchain, Bitcoin, Ethereum, Currency Exchange, Cross-Blockchain Transactions, Atomic Swap, Trading Platform, Instant Cryptocurrency Exchange, Data Analysis Tool, Process Analysis, Data Scraping, Heuristic Definition, Data Matching, Data Evaluation

Contents

List of Figures

List of Abbreviations

API	Application Programming Interface
BTC	Bitcoin
DDos	Denial of Service
ETH	Ether
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
LTC	Litecoin
OTC	Over-The-Counter
P2SH	Pay to Script Hash
REST	Representational State Transfer
RPC	Remote Procedure Call
URL	Uniform Resource Locator
USD	US Dollar
UTXO	Unspent Transaction Output

1 Introduction

1.1 Motivation

Since Bitcoin was introduced as the first digital currency in 2009, which provides a decentralized peer-to-peer system, the interest in this field has been increasing continuously (Nakamoto, 2008). Today this can especially be seen in the enormous rise of the market capitalization of Bitcoin in recent years and the number of different cryptocurrencies, which either just copy the Bitcoin protocol or try to improve it. Some have a structure that is technically similar to the Bitcoin system. For instance, Litecoin, which was launched in 2011, differs from Bitcoin in block creation time and the hashing function it uses (Narayanan, Bonneau, Felten, Miller, & Goldfeder, 2016). Other currencies like Ethereum, launched in 2013, offer new functionalities like the possibility of implementing smart contracts (Buterin, 2014). Every year many other cryptocurrencies are introduced. An analysis that was executed on 1,278 cryptocurrencies from 2013 to 2016 showed that more than the half are now extinct. Usually, the longer a cryptocurrency exists, the more unlikely it is to disappear (Lansky, 2016). The currently leading cryptocurrencies are Bitcoin with a market capitalization of more than 110 Billion USD and Ethereum with more than 38 Billion USD (Coinmarketcap, 2018). There are over 900 currencies listed on coinmarketcap.com, and the number is increasing (Coinmarketcap, 2018). Also, the interest in the technology behind the cryptocurrencies raised. The blockchain saves all ever-made transactions, and as it is decentralized, everybody can access the data written on it by joining the peer-to-peer network. Several explorer websites also offer a service to execute queries over the blockchain, without having to enter the network and give the user the possibility to perform simple queries. Although this data is public, the cryptocurrency systems aim to keep the user pseudonymous, e.g., by generating new addresses for every transaction (Moeser, 2013). Nevertheless, many tools have been developed in recent years, which use different algorithms and heuristics to not only generate general information about all transactions, like provided by the mentioned explorer websites but to get detailed information about the users of the system as well. Due to the popularity and massive usage of blockchains, especially for investment, the data hidden on the blockchain has become highly valuable to different stakeholders, such as state institution and banks, who want to uncover criminal activities, such as tax evasion. Therefore, many companies have specialized their business models to this field of research and offer analyses of the blockchain data, conducted by their specialized tools.

With the increasing number of cryptocurrencies, the demand for exchanging among them has also raised. Websites, like Bittrex or Bitfinex, offer their customers the possibility to exchange money between several currencies. These platforms are specially intended for trading. In the past years, new exchanger services have been introduced to offer a more comfortable and faster way for exchanging cryptocurrencies. These services are called instant cryptocurrency exchanges and are provided by companies like Shapeshift and Changelly. They don't execute trades on a closed system, like on trading platforms, which conduct transfers off-blockchain. Instead, they perform the transactions directly on the cryptocurrency networks. The data for every exchange is therefore written to the ledgers and can be retrieved from them. Every exchange generates two ordinary transactions on

two different blockchains: the deposit transaction realized by the customer and the withdrawal transaction sent by the exchange service. Although the data for both transactions can be retrieved from the corresponding ledgers, only the stakeholders, involved in the exchange, know about the link between these data.

1.2 Research Questions

Mostly, analysis of blockchain data is carried out over one specific blockchain. As interest in exchanging money between multiple currencies is rising, the aim of this thesis is the analysis and recognition of transactions involved in exchanges. Such transfers, which deal with transactions on different blockchains are also called cross-blockchain transactions.

First, we must analyze the processes of cross-blockchain transactions in detail. Based on this, we can build heuristics, which make it is possible to detect such transactions. After that, we implement a tool that analyzes the blockchain data of multiple cryptocurrencies and matches corresponding transactions. We then evaluate the outcome of this tool and finally provide it as an external service.

This thesis addresses three research questions, which guide through the implementation of the analysis tool:

RQ 1: What is the Current State of the Art regarding Cryptocurrency Exchange?

The first research question is dedicated to the literature research and helps to get an overview of the environment connected to the cryptocurrency exchanges. First, we must specify the available services related to these currencies. This listing includes the explorer platforms, the APIs for querying blockchain data and other services that interact with the blockchain. Furthermore, we categorize the different exchange platforms and explain their functionalities in detail. Also, we show which new ideas and prototypes regarding cross-blockchain transactions are currently developed and could be the future of cross-blockchain exchanges.

RQ2: How can Cross-Blockchain Transactions be recognized?

The second research question targets the design of the tools underlying algorithm. We must create heuristics, which make it possible to detect cross-blockchain transactions. The process requires to analyze the transfers of different exchange services in detail and to recognize patterns. Then it can be defined which parameters the algorithm needs to look at to detect these exchanges. These parameters can be for instance the time required for exchanging the currencies or the exchange rate. For this, we must also identify which data we can retrieve from the blockchain and which from external services off-blockchain. The underlying protocols and blockchain structures differ from currency to currency. The algorithm should take this into account and so be adaptable to different environments. Every exchange platform has a different exchange process and different parameters connected to it, e.g., transaction times, exchange rates and fees. Therefore, the recognition algorithm must be flexible and adjustable for different types of exchanges as well.

After the design of the algorithm, we can start the implementation process of the recognition tool, which also includes the connection to the required external services that provide the needed data.

RQ3: How accurate is the implemented Solution? What are the Limits?

The last research question takes a look at the quality of the implemented algorithm. We realize this evaluation by comparing the output with a set of data, which contains transactions that are correctly matched to an exchange. As there is no exchange data publicly accessible, the retrieval of such information is also part of this step. On the basis of the evaluation, we finally show the gains and limitations of the tool.

Possible applications for the implemented tool are the usage

- as an extern service for tracking single exchanges
- for analysis on top of the found data to generate general statistics about cross-blockchain transactions and to get transparency over the trading behavior
- for sophisticated analysis on top of the detected data, e.g., linking addresses over multiple blockchains to one user

1.3 Research Approach

We conduct this thesis as a design science research (Vaishnavi, Kuechler, & Petter, 2017). Therefore, we design and develop artifacts through the process and analyze their performance to improve them iteratively. The work will follow the steps suggested by the Design Science Research Process Model:

1. **Problem awareness:** A particular research field is examined to identify an unsolved problem, which can create valuable output if solved and leads to new knowledge. A proposal for the work is then realized. We mainly did this before starting the thesis to assert that the conduction of this work has additional value to the current knowledge base.

2. **Suggestion:** In this phase, a possible solution for the given problem is suggested. This includes designing new functionalities by creatively composing existing and new elements. In this work, this means building heuristics, with which we can recognize cross-blockchain transactions. Beforehand, we must determine multiple sources and methods to be able to declare correct heuristics.

3. **Development:** On top of the previously created algorithm, artifacts are developed, which execute the process for solving the research problem. The primary artifact in this thesis is the software program that implements the defined heuristics. We will show two different artifacts in this thesis which uncover exchanges in different ways differ in the external services they use.

4. **Evaluation:** The next step is to evaluate the artifact and the output of its execution according to previously defined criteria. For this, hypotheses about the outcoming

of the process are constructed. It must be proven if they are confirmed by the analysis and unexpected outcomes must be stated. In this work, the main aim is to evaluate if the implemented tool uncovers exchange data correctly. The knowledge and information gained in this process are then gathered and can be used in the suggestion phase of a new cycle to improve the artifacts design and its result. This is a valuable method for the implementation of the tool, as results gained from each execution of the tool help to improve it further.

5. **Conclusion:** Finally, the results of the conducted work and the knowledge gained from the process are recorded and shared. The aim is to improve the knowledge base and provide it to the environment, which consists of people, organizational and technical systems (Hevner, March, Park, & Ram, 2004). We will include in this result the degree of accuracy of the stated tool and the deviation from the expected outcome. The work can then be executed again or used for future research and improvements. The final outcoming of this phase is this thesis.

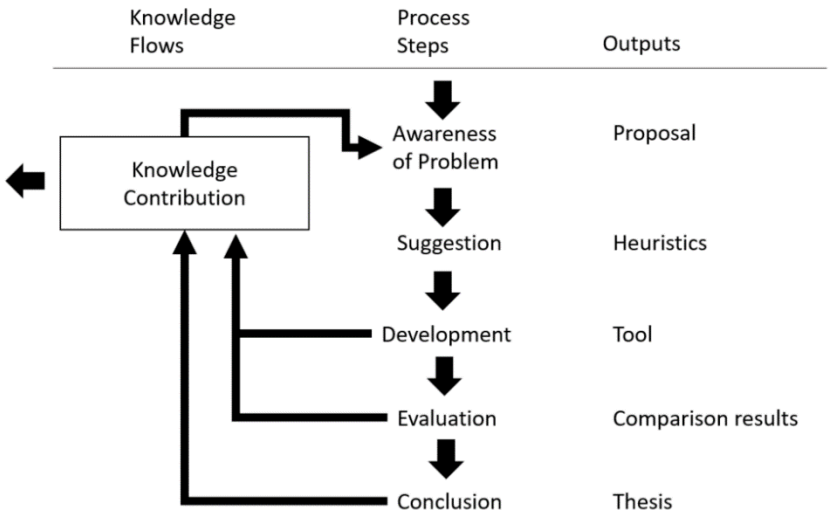


Figure 1: Design Science Research - Process Model (adapted from Vaishnavi, Kuechler, & Petter, 2017)

1.4 Overview

The structure of the thesis complies with the previously presented research questions and research method.

First, we answer the first research question in the second chapter by presenting and comparing different existing exchange services. In addition, we list and explain available tools and services for blockchain analysis. Furthermore, we introduce the terminology needed for a proper understanding of the explanations given in the conception.

The third chapter addresses the second research question. A part of the conception is the design of a scraper algorithm to retrieve exchange data from an exchange service that we need for the final evaluation. The other part deals with the design of the cross-blockchain

transactions identification algorithm. Here we explain which findings we retrieved from manual analysis of data from explorer platforms and the scraped data. Based on this, we construct and justify the heuristics.

The fourth chapter is a linking point between the second and third research question and shows how we implement the recognition algorithm. We explain the architecture and take a detailed look at the code of the tool.

We answer the last research question in the fifth chapter, which describes the outcome of the comparison of the tools data with the data retrieved by the scraper algorithm.

Finally, we summarize the work and outcome in the sixth chapter and draw a conclusion. In addition, we give an outlook for further development of the recognition tool.

2 Background

Throughout the years an ecosystem of services around the cryptocurrencies has arisen. Nevertheless, in this thesis, we will focus on two types of service groups. Firstly, we will show the available exchange services on the market, classify them and analyze their exchange processes in detail. The understanding of the process will be necessary for creating heuristics for the recognition algorithm. Moreover, we will look at the services and tools available for analyzing the transaction data on the blockchains.

2.1 Exchange Services

The high interest in exchanging different cryptocurrencies is satisfied by many different exchange services. These companies enable trade between several digital currencies. Some also include exchanging conventional fiat money. The exchange services can be divided into three different categories. Firstly, there are trading platforms, in which users can register and transfer money through a central authority. This authority mostly makes profits by charging fees. As trades are executed on a closed platform and no transaction is sent to the blockchain networks, there is no data saved on the ledgers. The uncovering and matching of customer addresses are therefore not possible. Furthermore, there are over-the-counter markets, which do not use a centralized mechanism to match bids with offers. Instead, exchanges are performed directly between two parties. A third party is only involved in helping to find corresponding partners, which want to exchange the currencies at a self-defined rate both agreed upon. The last category contains the instant cryptocurrency exchanges, which are similar to the OTC exchanges, but in which the exchanger takes the role of the counterparty. In the following, we will examine this kind of exchange services in detail and compare them with each other. Finally, we will show how exchanges might be handled in future in a decentralized manner. (Galitskiy, Shpin, & Virk, 2015)

2.1.1 Trading Platforms

Overview

Currently, there are over 200 trading platforms listed on Coinmarketcap (April 2018). Exchanges with one of the highest market volumes are Binance, Bitfinex and Bittrex, all with a trading volume exceeding multiple million USD per day. Every platform allows trading with a limited number of currencies, including cryptocurrencies as well as fiat money. Coinmarketcap also lists the most traded currencies on these platforms. These are among others Bitcoin and Ethereum, Bitcoin Cash and Litecoin (Coinmarketcap, 2018).

Characteristics

Trading platforms are websites, to which users can get access by logging in. Mostly this requires identification with an identification card or passport, the verification of a credit

card and provision of personal data. This process can take time and creates a barrier for those who want to do single exchanges and don't have the intention of trading frequently.

Moreover, the user interface of a trading platform is aligned to the needs of people that trade and therefore offers many possibilities of interaction. It might be unsuitable for doing a single or a small number of exchanges, as the platform includes complex diagrams, bid/ask orders and other features.

In the beginning, the user must transfer money to his account or wallet to be able to trade. This means the trading service holds all funds of their customers. On the one hand, this is an advantage as the centralized mediator erase the risk of fraud by the counterparty. The user can be sure to get the exchanged money back and trust to the counterparty is not needed. On the other hand, this can be a risk for the users as their assets can get lost in case the company goes bankrupt or gets hacked. An example of such an incident is the bankruptcy of the Bitcoin exchange Mt. Gox in 2014, after over 700,000 Bitcoins were stolen. Back then this was the leading exchanger, responsible for more than 70% of all Bitcoin transactions (Frunza, 2015). The probability of failure is extraordinary high. From 2010 to 2013, 45% of bitcoin currency exchanges were closed. (Moore, & Christin, 2013)

Trading fees are very low reaching from 0% to 1% depending on the chosen platform (Bitcoinwiki, 2018). Additionally, the customer must pay fees when transferring money to or out of the trading platform wallet.

All in all, these platforms are intended for people that trade and execute exchanges frequently. Although they offer low fees and minimize the risk of fraud, they are not suitable for simple, one-time transfers as the user must go through many steps to finally receive the exchanged amount on his cryptocurrency wallet.

2.1.2 Over-The-Counter Markets (OTC)

Overview

In comparison to trading platforms, OTC transactions are not performed on a closed platform. The exchange itself happens between two parties, and since it is executed directly, transfers are written to the blockchains. A third party is only involved in finding appropriate partners for exchange. Therefore, the third party mostly doesn't hold the assets of the traders on their accounts. Over-the-counter transactions are made either on such peer-to-peer marketplaces or directly between two people without any intermediary.

Examples of peer-to-peer marketplaces are services like Bitcoin-otc (Bitcoin-otc, 2018) and ItBit (ItBit, 2018). Traders can make offers on this platforms and search for appropriate buyers.

Characteristics

According to (Galitskiy, Shpin, & Virk, 2015), there are several advantages compared to trading platforms using OTC transactions. These are stated in the following.

First, the exchange of a high amount of money on a trading platform can lead to a price slippage at this specific exchange. As OTC transactions are not executed on a closed platform, this risk can be bypassed. Due to this, high-volume exchanges are preferably performed as OTC exchanges in case the volume of a trading market is relatively small (Galitskiy, Shpin, & Virk, 2015).

As mentioned, trading platforms require a lot of personal information from the user for registration. Some OTC exchanges, which help to find a partner for trading, don't have such regulations and offer a better choice for people that want to stay anonymous.

In contrast to trading platforms, OTC markets don't hold assets of their users at all. Therefore, there is no risk for the customer of losing a significant amount of money in case a company holding funds goes bankrupt or gets hacked.

But there are also some downsides of OTC markets. These also can be different at every OTC market.

At first, there is a risk of fraud by the counterparty. As there is no centralized authority, a trader can never be sure to get back the agreed exchange money. On some peer-to-peer markets, the marketplace supplier tries to solve this problem by offering a reputation system and so creating trust between the counterparties.

Another downside of OTC markets is that some providers might not have enough liquidity. Hence, exchange with high amounts can't be fulfilled.

There are services which claim to be an OTC market but don't fulfill all of these characteristics. Services like LocalBitcoin (LocalBitcoin, 2018) and Bitquick (Bitquick, 2018) match two suitable traders, which are willing to exchange assets with each other, just the same way as on an OTC market. The difference is that they hold assets to guarantee the customers more security. Nevertheless, the money is only deposited by the customer for a short time for a single transaction. LocalBitcoin, e.g., forces one trader to store his Bitcoins needed for the exchange to the account of the company. Then, the other trader sends the corresponding asset directly to the counterparty. When this transaction is confirmed the service finally transfers the stored Bitcoins to the other trader.

2.1.3 Instant Cryptocurrency Exchanges

Overview

A new form of service exists that provides a fast and easy exchange process: the instant cryptocurrency exchange. Just as at the OTC markets, instant cryptocurrency exchanges are not executed on a centralized platform and transactions involved in the exchanges are directly saved on the blockchains. Also, the transfers are done between two parties. The difference is that the service takes the role of the counterparty. Examples are Shapeshift, Changelly, Evercoin, Blocktrades, and CoinSwitch. The most popular instant exchange

services currently are Shapeshift and Changelly. Therefore, we analyze them more in detail later.

Characteristics

Just as the OTC markets, these services do not require strict identity verification when exchanging cryptocurrencies. They provide a fast and short exchange process without having to reveal one's own identity.

Similar to the trading platforms, the service holds client funds on their accounts, as the user must deposit money to start an exchange. Nevertheless, this applies only to single transactions and for the time the trade is executed. Hence, the risk of losing high amounts of money in case the company losses funds is very low for the customer.

Instant cryptocurrency exchanges furthermore provide competitive exchange rates, as the best price is retrieved comparing several trading platforms.

Also, they usually allow more different currencies for exchanging than trading platforms, which have a limited offer and don't include some coins at all. The offerings on instant cryptocurrency exchanges are wider spread as they interact with multiple trading platforms. Therefore, the user doesn't have to register himself on many different trading platforms.

Since the transaction data of exchanges executed by an instant cryptocurrency exchange is directly saved on the blockchains and is publicly accessible, this kind of service is suitable for the cross-blockchain transaction analysis. Moreover, they have a high transaction volume and are visibly growing in number. Therefore, our analysis focuses on exchanges of these services.

2.1.4 Services without Centralized Intermediaries

Currently, there are many solutions in development to improve cross-blockchain transactions and which may be the future step for exchanging cryptocurrencies. Nowadays, exchanging digital currencies is slow, requires trusting a counterparty and has high costs. In the following, we briefly present some concepts, which try to solve the given problems.

Atomic Swaps

A promising concept for exchanging currencies between different blockchains without the need of trusting an intermediary is called atomic swap. The main feature of this is the security that either both transactions of an exchange happen or neither of them. First, the two parties, which are involved in the trade, must agree upon an exchange rate. Then both submit their transactions using hashed time-locked contracts (HTLC). These contracts guarantee that both partners can claim a refund in case something has gone wrong during

the exchange. The underlying algorithm contains multiple steps which must be executed manually by the exchange partners. That is why various organizations have started implementing protocols to facilitate this process. The most known ones are listed below. (Herlihy, 2018)

Lightning Network

The Lightning Network is a decentralized network in which peers can instantly execute payments among each other and in which off-chain smart contracts guarantee transaction security. Currently, the implementation is adaptable to Bitcoin and Litecoin blockchains. This network was specially designed to decrease the transaction load on blockchain networks, to allow faster payments and decline transaction fees. As it is handled off-blockchain, the number of transactions is not limited by the blockchain protocol (Bitcoin blocksize is 1 MB) and so millions of transactions can be executed at the same time. Furthermore, no block creation times (for Bitcoin approximately 10 minutes) must be kept in mind and transactions are fulfilled in seconds. Furthermore, transfers can be made at low costs, as blockchain network fees don't have to be paid for every transaction. This feature allows the user to send small amounts of money. To make this possible, the protocol creates a ledger entry between two peers, which holds a given amount of money from both parties. All transactions between two peers are documented on this 2-of-2 multisig address, which means that both participants must sign every change in balance with their private key. The history of transactions is therefore recorded off-blockchain. Either of the peers can write this entry to the ledger and release the funds at any time, which then creates a single transaction on the blockchain. As the protocol creates a whole network consisting of such addresses a payment channel doesn't have to be created between every peer. Instead, payments can be passed through multiple peers on this network. As long as blockchains use similar hash functions, the Lightning Network can execute cross-blockchain transactions between them, allowing a participant to send money to destinations on other ledgers. This feature was yet tested transferring money between the Bitcoin and the Litecoin network. (Poon, & Dryja, 2016)

Interledger

The Interledger protocol aims to make payments across different systems possible. Just as in the Lightning Network, money is sent through multiple peers. In the case of Interledger, this allows a participant to send and receive money from different ledgers without having to create accounts for each of them. The Interledger protocol is an open architecture and therefore allows to integrate any payment system, e.g., distributed ledgers or banks. Participants can send any kind of asset through the network, including cryptocurrencies, stocks, and commodities. For this, the ledgers are connected by hubs, which receive payments from one or multiple ledgers and send an amount of the desired asset to one or multiple other systems. If the transaction doesn't happen within one single system, the connector can determine an own exchange rate for exchanging an asset to a different one and so generate revenue from the exchange difference. As the participant can request and compare the exchange rate of multiple connectors, the fee for exchanging

is low. The protocol also ensures that the money sent cannot be stolen during the transaction process.

COMIT Network

The COMIT (Cryptographically-secure Off-chain Multi-asset Instant Transaction) Network has a similar concept as the Interledger protocol. It compares itself with the TCP/IP protocol which once connected multiple local networks to a single network, the internet. COMIT has the aim to implement a protocol for exchanging assets between various blockchains, just as the internet did it for transferring information. The cross-chain routing protocol (CRP) provides a secure, low cost and fast network for cross-blockchain transactions using off-blockchain smart contracts to build a cryptographically secure and trustless network. The connectors between multiple blockchains are called liquidity providers. They provide the assets needed for transferring the money from one participant to another one. Anyone can take over the role of a liquidity provider, such as banks, exchanges or private persons. (Hosp, Hoenisch, & Kittiwongsunthorn, 2018)

2.2 Examples for Instant Cryptocurrency Exchanges

In this chapter, we will take a closer look at the process and the conditions of two instant cryptocurrency exchanges to get an initial understanding of their exchange process.

The two exchange services we will focus on (Changelly and Shapeshift) have similar exchange process, which we show in Figure 2. First, the user sends the currency he wants to exchange from his cryptocurrency wallet to the wallet of the exchanger, paying transaction fees on the involved blockchain. The exchanger then determines an exchange rate and an exchange fee he will charge and calculates a final amount of the currency the user inquired. He sends this amount from his wallet to the wallet of the user, also paying transaction fees on the blockchain network of the withdrawal currency. We see that an exchange generates two transactions on two different blockchains. In Figure 2 for instance, the first transaction transfers 1 BTC from the user to the service and the second the corresponding amount of 14.6 ETH from the service to the user. In the following, we describe the procedures of each service more in detail.

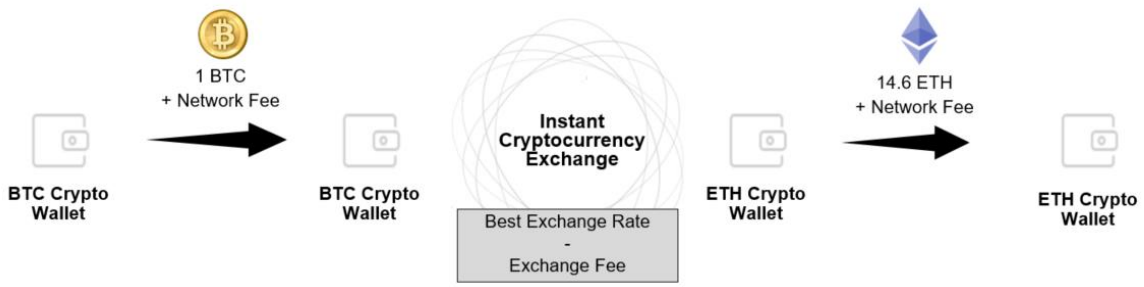


Figure 2: Instant Cryptocurrency Exchangers – Exchange Process (Adapted from Changelly, 2018)

Changelly

The Bitcoin mining pool MinerGate founded the exchange service Changelly, and released it in 2015. It counts more than 3 million users on the website every month. Currently (April 2018), it is possible to exchange over 50 different currencies, including fiat money (USD and Euro), which requires the usage of a credit card. The service can be used directly on the website or externally by other services, as it provides an API for executing exchanges. For instance, cryptocurrency wallet providers like Jaxx or Coinami integrated it to allow their customers to shift money between different cryptocurrency wallets. (Changelly, 2018)

The exchange process works as follows:

First, the user must login or create an account, which includes specifying an email. The account allows the users to see a history of all exchanges made on the platform. Protecting the account with a 2-factor authentication is possible.

Next, the user can choose which two currencies out of the available ones he wants to exchange and sets the desired amount (Figure 3). This amount must be high enough to cover transaction fees of the blockchains. A maximal amount is not existent. The expected outcome is immediately shown, as well as additional transaction information that includes:

- The exchange rate, which is generated by searching for the best rate from different trading platforms like Poloniex, and Bittrex.
- The exchange fee, which is 0.5% of the amount of the withdrawal currency. The network transaction fee, which the exchanger must pay to send the money, is additionally subtracted from the final amount. The user must be aware that he also must pay a transaction fee by sending the money to the exchanger and this should be counted as costs for the exchange too.
- The time the transfer will take. Changelly states that most exchanges take 5 to 30 minutes. This mainly depends on the time it takes to confirm the transactions on both blockchains. The confirmation time is influenced by factors like the general block creation time, the height of transaction fees set and the current blockchain load. DDoS attacks and updates on the website of the exchanger can additionally enlarge the waiting time.

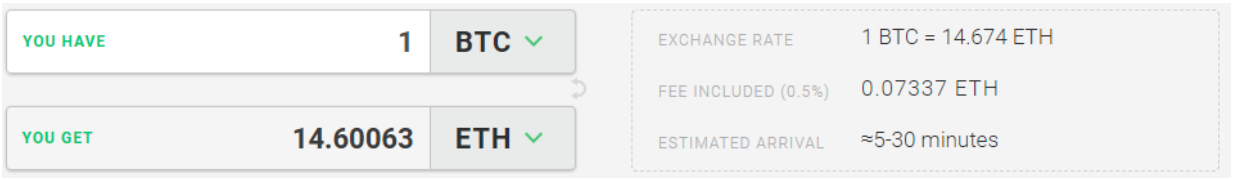


Figure 3: Changelly Exchange Details (Changelly, 2018)

The next step is to specify the wallet address of the recipient of the exchanged money and confirm the exchange, which generates a transaction id. This id allows tracking the status

of the exchange. Then the wallet address of the exchanger is shown, to which the customer must send the defined amount of money. For every trade, a Changelly generates a new address. Nevertheless, it is also possible to reuse an address. So, if money is sent again to this address, a new exchange is executed, and the output is sent to the same output wallet as in the previous exchange. After depositing the money, the exchanger waits to receive it. This happens when the corresponding transaction is added to a block and saved on the blockchain. Changelly then calculates the exchange outcome again, as cryptocurrencies are very volatile and rates probably changed in the bygone time. After sending the withdrawal to the user, details of the exchange are provided, which include the outcome and the hash of the transaction executed by the exchanger. This information serves as a proof that the trade was completed successfully.

Shapeshift

The Shapeshift exchange service is available since August 2014 and allows exchanging between more than 40 different cryptocurrencies. Transactions with fiat money are not possible and won't be provided in future, according to the Shapeshifts policy. Due to several attacks and loss of money, the infrastructure of the service has been changed in 2016 (Leung, 2016). The service can be used on the main website or externally through an API. Cryptocurrency wallet providers like Exodus, Jaxx, Coinoni, and myEtherWallet include the service in their applications. The company is more open regarding showing data of exchanges executed on their platform. They, e.g., provide general information in real time on its official website. Therefore, it is possible to extract the number of transactions per day and the daily trading volume. In April 2018 this value mostly lied under 1,000 BTC. It is low in comparison to conventional trading platforms. Leading ones had a trading volume of over 100,000 BTC daily in April 2018 (Coinmarketcap.com). The statistics also show that the most popular exchange is between Bitcoin and Ethereum. Furthermore, it is possible to get anonymized data of single exchanges from the API. This circumstance will be exploited later to build a scraper that retrieves data from Shapeshift. (Shapeshift, 2018)

The exact exchange process follows these steps:

At the beginning, the user can choose if he wants to execute a quick or a precise exchange (Figure 4). For the quick exchange, no exchange amount must be specified. The user can send any amount to the exchanger as long as it is in the defined deposit limits. Furthermore, it is possible to set the option allowing a deposit address to be reusable. A precise exchange doesn't have this option and requires defining an exact amount. This can be either the amount the user wants to send or the amount he wants to receive. The corresponding amount is calculated immediately. Having the possibility of setting the outcome amount (which is not possible in Changelly) is especially convenient if the user doesn't want to swap and send the money to his wallet, but to someone else's wallet and has to pay an exact amount. He can then send money in a specific currency for which he doesn't even own a wallet. The company states that it provides a better exchange rate and a higher deposit limit for a precise exchange. While setting up an exchange, additional information is provided, which includes:

- The exchange rate, which is influenced by different values. For instance, the rates of external exchange platforms, from which Shapeshift retrieves its money, and the current property of cryptocurrencies of Shapeshift.
- The minimum and the maximum limit of the deposit. If the user sends amounts beyond these limits, the money is sent back to him subtracting the transaction fee. The lower bound guarantees that transactions are above the transaction fee paid to the network and the upper bound prevents too big exchanges, which could cause price slippage at extern exchanges and so diminish the rate. These limits allow Shapeshift to offer constant rates for different exchange amounts.
- The "miner fee" is the only fee Shapeshift takes for providing their service. Therefore, for every exchange, this fee is subtracted from the outgoing amount, which was calculated with the best rate. For every currency, the service claims a different miner fee, which applies for every exchange, regardless of the amount involved. The company occasionally adjusts this fee according to the current market conditions. As Shapeshift still pays the transaction fee to the blockchain network for sending the money to the user, their total profit can be calculated by subtracting this transaction fee from the "miner fee".

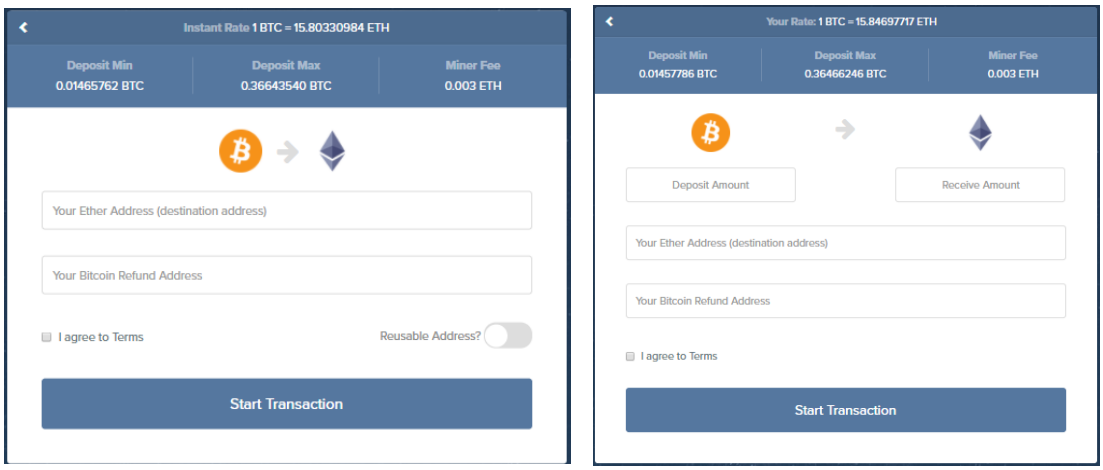


Figure 4: Shapeshift quick exchange (left) and precise exchange (right) (Shapeshift, 2018)

In both types of exchanges, the user must specify the address where he wants to get the exchanged money to and a refund address in case something goes wrong, and the deposit must be sent back. After agreeing to the terms, the user can start the transaction.

The address where the user must deposit the money is shown. For the precise exchange, the deposit must be sent to the network within the time of 5 minutes, but not be confirmed yet. For small amounts, Shapeshift only waits until the transaction is written to the blockchain within a block. For larger amounts, it waits for one or more confirmations. After receiving the money, the exchange rate is calculated again. Therefore, the outcome can differ from the first prediction. After the exchanged amount is sent to the user, he can see detailed information about the exchange, including the final rate and a hash of the transaction.

Other Instant Exchange Services

Besides from Shapeshift and Changelly, which are currently the most used instant cryptocurrency exchanges, other new companies are providing a similar service. These will be briefly described below.

Blocktrades.us, for instance, allows instant exchanges between 7 different cryptocurrencies and finds the best rate by analyzing more than hundred market paths. An account is not needed, and an API for integration in external applications is provided. Fees are not transparent but are included directly in the outcoming amount. A deposit limit must also be considered. (Blocktrades.us, 2018)

The exchange service from Silicon Valley named Evercoin provides their service on a website, as well as on mobile platforms (IOS and Android). Additionally, it can be used through an API. Exchanges are executed instantly between 20 different cryptocurrencies without having to log in. All fees are already included in the estimated outcome that is shown when a trade is requested. Exchanges can only be done within certain limits. One confirmation is required to process a deposit and exchange the currency. (Evercoin, 2018)

Coinswitch compares different exchange services on their website. They include trading platforms like Cryptopia and Bittrex, and instant cryptocurrency exchangers like Shapeshift, Changelly, Blocktrades and their own exchange service. For every requested exchange, the user gets an overview, showing which platforms provide the desired currencies and which rate they offer. He can then choose the best option and execute the exchange without leaving the website, as requests for all platforms are integrated using the available APIs. CoinSwitch itself doesn't provide any API. More than 275 cryptocurrencies are available for exchange. Apart from the fees from the external exchangers, CoinSwitch charges no additional fee using the instant exchangers and 0.25% - 0.98% using the trading platforms. (Coinswitch, 2018)

2.3 Blockchain Analysis Tools

In this chapter, we present tools used for blockchain analysis. We categorize them in basic tools for retrieving blockchain data, open source analysis tools, and paid analysis services. As stated before, Bitcoin and Ethereum currently have the highest market capitalization, are two of the most traded currencies on the trading platforms, and the exchange between these currencies is the most popular at the Shapeshift exchange. Therefore, we will focus on tools for analyzing these two cryptocurrencies.

2.3.1 Blockchain Data Retrieval

Full Nodes

Due to the distributed character of the blockchain, it is possible to access all ever-done transactions by downloading the whole ledger. This is achieved by instantiating a node,

which communicates with the network. Light versions of nodes are provided, if participants only need a limited set of interactions with the network, such as sending transactions. This kind of nodes only requires downloading a part of the blockchain (Duong, Chepurnoy, & Zhou, 2018). Nevertheless, for the analysis of every single transaction, a node containing all blockchain data is needed. For Bitcoin, this would be a full node, currently requiring more than 160 GB (April 2018) of free disk space (Blockchain.io Statistics, 2018). An Ethereum node synchronized with the Geth client, which downloads the entire blockchain data but prunes old states, needs almost 70 GB (April 2018) of disk space (Etherscan.io, 2018). Such nodes guarantee that all downloaded blocks and transaction are valid and follow the blockchains consensus rules. The local node provides an API, allowing interactions with the network over JSON-RPC calls. Besides the methods needed for performing transactions, also requests can be made to get information about the data on the blockchain and from the network. For instance, we can query blocks, transactions, and addresses. There are various clients provided that implement this protocol and make it possible to interact with the network using different programming languages. Both blockchains, e.g., offer client implementations in Python, Javascript, Go, C++ and other languages (Ethereum Github, 2018). Also, blockchain data is stored locally on files and can be retrieved from these directly. For Bitcoin, for instance, concatenated raw blocks are stored in .dat files in the main directory (Alqassem, & Svetinovic, 2014). As many API clients are available, the programming language for building an analysis tool is flexible.

Cryptocurrency Explorers & APIs

Setting up a full node requires much disk space and time. Therefore, many third parties offer a service to do the queries on their full node. The so-called cryptocurrency explorer websites show information about blocks, transactions, and addresses of different currencies. Some also provide additional data, e.g., about the network, known addresses, and general statistics. Besides of only making requests on their websites, many of these explorers also offer an API. In this way, the data can be retrieved via HTTP calls programmatically. The downside is that some of these services have request limits and retrieving data over HTTP request is slower than reading it from an own node. Nevertheless, it can be suitable for analysis on a limited number of blocks and allow more extensive analysis by exposing additional data not retrievable from a node.

One of the most popular explorers for Bitcoin is blockchain.info. The Website provides information about all blocks and addresses, shows charts and statistics, and offers a bitcoin wallet. Data can be retrieved in JSON format through an API. The additional data is valuable for the exploration of exchanges. For instance, the API returns the time a transaction was sent to the network. Whereas the response of a full node only contains the time the block, in which the transaction is included, was written to the blockchain. (Blockchain.io, 2018)

Popular explorers for Ethereum are etherscan.io and etherchain.org. All blockchain data can be retrieved through the web interface. Nevertheless, they offer APIs with a low request limit. A better solution is Infura, a service which provides a scalable blockchain

infrastructure and fast access to the Ethereum transaction data. The service is not an explorer but is specialized in providing access to the Ethereum network without having to set up an own node. It can handle a high number of requests, currently more than 2 billion per day. Therefore, it is also suitable for use in the analysis field. The API can be addressed through simple HTTP calls. (Infura.io, 2018)

2.3.2 Open Source Analysis Tools

Many open source tools related to blockchain data analysis, analyze and visualize general market data of different cryptocurrencies. More sophisticated tools, which execute complex queries and algorithms, are rare and mostly outdated. Examples are tools like BitcoinVisualizer (BitcoinVisualizer, 2018), BTCSpark (BTCSpark, 2018) and BitIodine (BitIodine, 2018), which weren't updated for several years. They allowed running complex processes on the Bitcoin blockchain, like, e.g., clustering addresses and revealing connections between users.

A recent open source project is BlockSci. This tool was published by the Princeton University and is specialized on analysis of Bitcoin blockchain data. Also, it supports the Bitcoin Cash, Litecoin, Namecoin, Dash and ZCash blockchains. It was specially designed for analysis and proves to be faster than tools implemented so far, also due to a parser which restructures blockchain data and saves it to an analytical in-memory database. Additionally, it retrieves exchange rates and records data from the blockchain network and so allows a wide range of different queries. The use of the tool is mainly realized in Python, but C++ can also be utilized to get better performance. With this tool sophisticated analysis tasks can be performed, such as assessing the privacy of cryptocurrency transaction by linking and clustering addresses. (Kalodner, Goldfeder, Chator, Möser, & Narayanan, 2017)

Besides BlockSci, no comparable tools for blockchain data analysis are provided on open source. Especially in the field of cross-blockchain transactions, which include multiple blockchains, no analysis effort was done yet.

2.3.3 Commercial Analysis Tools

It is visible that analysis of blockchain data is more present in the business field. The growing interest of blockchain-based technologies led to its expansion into many different industries and areas. Because of this increase of usage, more organizations have the interest in accessing and analyzing the data saved on the ledger. Therefore, companies specialized in such kind of services have been established in the past few years.

One of the leading companies in this field is Chainalysis, which was founded in 2014. The analysis is exclusively executed on the Bitcoin blockchain. The company offers its service to three different customer fields, which are financial institutions, businesses doing Bitcoin transactions and law enforcement agencies fighting against cyber threats. Chainalysis provides reports and visualizations of customer activities to make the assessment of business risks possible. Furthermore, it tracks activities of addresses and

provides the data through a real-time API. The activity tracking is also used to convict criminals, which extort or launder money. (Chainalysis, 2018)

Another company providing services in this field is Ellitic. Its blockchain analysis tools are also limited to Bitcoin data and offered to law enforcement agencies and financial institutions. The primary task of the company is the sophisticated analysis of Bitcoin transactions, with which it is possible to uncover criminal activities and link addresses to real identities. This additional information gives companies transparency over Bitcoin accounts and allows them to reduce their risk when handling with specific customers. (Elliptic, 2018)

2.4 Terminology

In this chapter, we will take a closer look at the structure of blockchain protocols and the process of doing a cryptocurrency transaction. This terminology is needed to understand cross-blockchain transactions in detail, as well as the methods used to recognize them. As the focus in this thesis is on exchanges between Ether and Bitcoin, these technologies will be explained.

Cross-blockchain Transaction

First, we want to define what exactly a cross-blockchain transaction is. It is an exchange between two participants, which involves multiple cryptocurrencies. For this kind of transfers, we must differentiate, if they are executed off-blockchain or on-blockchain.

On-chain exchanges always involve two normal blockchain transactions on two different blockchain networks. We assume that participant A wants to exchange 10 ETH for 1 BTC with participant B. A then sends 10 ETH from his Ethereum address to Bs Ethereum address, and B sends 1 BTC from his Bitcoin wallet to the Bitcoin wallet of A. OTC, instant cryptocurrency exchanges and atomic swaps follow this structure and are conducted on-chain. The benefit of on-chain transactions for analysis is that both are written to the blockchain and are publicly visible. Nevertheless, only the involved participants are aware of the connection between these two transactions.

Exchanges, involving off-chain transactions, happen on external platforms and are only tracked there, thus are not visible on the blockchain. Trades that are executed on trading platforms have this characteristic. All the customer money is held on the accounts of the trading platform. All funds on the platform are a representation of this money. Hence, participants can execute multiple exchanges without saving any data on the blockchains. This principle also applies to the previously presented protocols, which create payment channels between users, and thus allow to execute exchanges off-chain. Nevertheless, every off-chain transaction is connected to on-chain transactions, as every channel, containing multiple off-chain transactions, is initiated with a deposit, which is an on-chain transaction, and closed by a withdrawal, represented by another on-chain transaction.

Blocks

A blockchain consists of multiple blocks which are linked to each other over one-way hashes. Every block has only one predecessor and contains a hash, which is generated from the transactions included in the previous block. The chain can't be modified unnoticedly and therefore is immutable. Every block contains multiple transactions. These transactions, which are sent by different participants of the network, are collected by various miners, who populate blocks with these transactions and try to append their block to the chain. (Narayanan, Bonneau, Felten, Miller, & Goldfeder, 2016)

In the Bitcoin network, a new block is generated approximately every 10 minutes. The size of a block is limited to 1 MB and can contain up to 1,978 transactions, assuming the average transaction size of 530 Bytes. So, 3.3 transactions are confirmed per second. (Ploom, 2016)

In the Ethereum network blocks are generated much faster. They are appended about every 14 seconds ("Ethereum Average Block Time", 2018). An Ethereum block has no size limitation, but a maximum Gas limit per block, which is adjusted over time. Gas is the fee paid for every transaction. The gas paid for all transactions contained in a block can't exceed the defined limit. The current Gas block limit is around 8 Million ("Ethereum Average Gas Limit", 2018). Considering the standard Gas limit for a transaction of 21,000 (Wood, 2014), a block can contain up to about 380 transactions. This corresponds to about 31.7 transactions per second.

Transactions

Transactions on a blockchain represent the transfer of value between addresses, which belong to different owners. Each transaction has one or multiple inputs and one or multiple outputs, each showing the addresses involved.

A Bitcoin transaction can contain multiple inputs, as well as multiple outputs. Inputs are always unspent transaction outputs (UTXO). They are values, which were received by an address and were not spent yet. An UTXO must be spent entirely in only one transaction. If a user wants to send an amount of money, which is smaller than the value of his UTXO, the desired amount of this UTXO is sent to the recipient and the rest to a change address, which is owned by the sender. To transfer high amounts, multiple UTXO can be combined. As soon as a UTXO is used in a transaction, it can't be used anymore in other ones. All outputs become UTXO, which can be spent on future transfers. The sender must pay a fee for each transaction. The height of the fee influences how fast the transaction will be included in a block. (Buterin, 2014)

In Ethereum there is no concept of UTXOs. Therefore, no change addresses are needed. Every transaction has one input address and one output address. An address can be seen as an account. It stores all values sent to it, and output values can be defined independently from the received inputs. For each transaction, the sender must pay a fee, which is calculated by multiplying the set gas with the gas price. If the fee is too small, the transaction may be not included in a block. (Buterin, 2014)

Summarizing, the difference between both currencies is that in Bitcoin there are no accounts and users can spend the money from multiple addresses at once in one transaction. In Ethereum one address represents one account and thus only the funds of one address can be involved in one transaction. This fact will be necessary for the analysis of the processes the exchangers perform on these two networks.

Another important differentiation for the analysis is the existence of two timestamps involved in every transaction. First, there is the transaction time, which is the time a transaction was sent to the network by the corresponding sender. As nodes on the network don't receive this transaction at exactly the same time the transaction time of different nodes can slightly differ. Secondly, there is the block confirmation time that is the time the block, which includes a transaction, was appended to the blockchain ("Bitcoin confirmation", 2018). Transaction time and block confirmation time can lay wide apart, depending on the height of the set fee or the network utilization. Especially for Bitcoin transactions, this can be a high range, as blocks are confirmed only about every 10 minutes. For Ethereum transactions this is not as relevant, because blocks are confirmed every couple of seconds, and the throughput is significantly higher than on the Bitcoin network.

3 Conception

3.1 Approach

In this chapter, we go step by step through the process, which was taken to develop a tool that can recognize cross-blockchain transactions. First, we look at the possibilities that are given to retrieve data from exchange services. This data is needed to understand the exchange process in detail. Based on this, it is possible to define heuristics that allow us to implement a recognition algorithm. After this, we explain how we can evaluate the output. Finally, we describe how the tool can be provided as a service.

3.2 Scraping of Exchange Data

In the beginning, we encounter the problem that we need a set of exchange data from exchange platforms, with which we can evaluate the output of the implemented recognition tool. Established trading platforms like Bitfinex (“Bitfinex API”, 2018) and Bittrex (“Bittrex API”, 2018) mostly offer an API with which it is possible to retrieve exchange data. The set includes detailed information, like the quantity of exchanged currencies, the exact exchange price and the exchange time. Also, general statistics like the transferred volume in a certain period are public. Coinmarketcap.com sums up such information for a big number of traders on their website (“Exchange Volumes”, 2018). Instant exchange platforms in comparison, only offer a limited view into their processes. There is no detailed exchange data available on the internet. The attempt of requesting data by directly contacting the companies was also unsuccessful. The reason could be that instant exchange services want to keep the data of their customers as private as possible, which is also one of the advantages they claim to have in comparison to traditional trading platforms. Furthermore, exchanges of trading platforms happen off-blockchain and therefore cannot be connected to blockchain addresses or even identities by using the available data. This is not the case for instant exchanges, as the involved transactions are directly visible on the blockchains. Revealing data like currencies and amounts of an exchange would make it possible to link exchanges to accounts.

3.2.1 Analysis of Data Exposure

The first step is to check if the exchange services expose data at all and which exchange data can be made visible. Therefore, we analyze every service in detail.

Changelly

The only way to get data from Changelly is from their Instant exchange API (“Changelly API”, 2018). An API key is required to use it. The requests are categorized into three different classes: requests for quotation, generating transaction and providing transaction status and history. We are only interested in the first and last categories, as these could expose exchange data. The first class includes methods for retrieving general

data like getting a list of all available currencies, the minimum exchange amount for a given currency and the estimated outcome for a given amount of money. The other request class includes a method for getting transaction data by a Changelly related transaction ID which is returned after generating an exchange and a method for showing a list of all transactions. The history only includes the transactions related to the account, with which the API key was created. Therefore, there is no possibility to get a general set of exchange data from Changelly.

Shapeshift

Shapeshift offers more insights into their service. This is already visible on the main page, on which the most recent exchanges are shown, with the corresponding currencies and amount transferred. Additionally, statistics for the last 24 hours are presented, including the number of transactions, the exchange volume in Bitcoin, the average processing time and the most popular exchange (Figure 5). Just as Changelly, an API is offered to interact with the service ("Shapeshift API", 2018). The API allows GET and POST requests. The POST calls are used to create and cancel transactions or for requesting an email receipt. With the GET calls, general data including the supported currencies, rates, fees and exchange limits can be retrieved. Furthermore, exchange information is exposed. Thus, the time left for sending the money or the status of a transaction can be checked passing only the deposit address to the call. It is also possible to create a private API key and request a list of own exchanges. Finally, the recently executed exchanges, which are also shown on the main page can be retrieved. As most of the requests don't require a private API key, data can be scraped to some extent. Three requests were identified to be useful for the scraping and are described in detail.

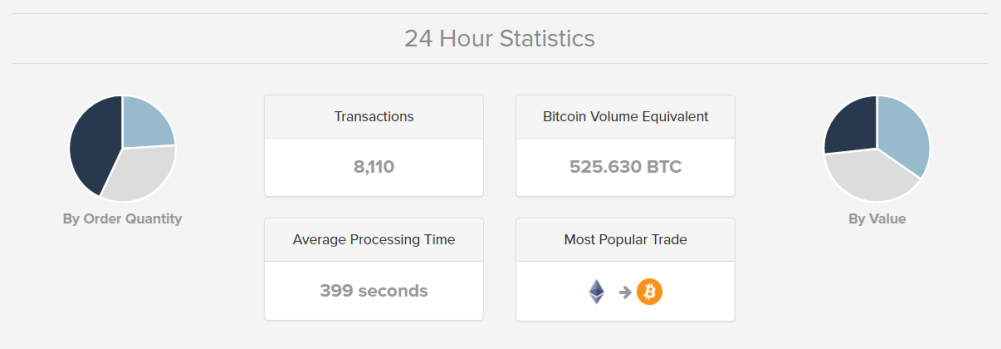


Figure 5: Shapeshift Statistics (24.03.2018) (Shapeshift, 2018)

First, the fixed Shapeshift exchange fee, called "miner fee", can be retrieved for all currency by the "Market Info" request. The response lists all currency pairs and the fees the service asks for each of these. Unlike Changelly, Shapeshift charges the same amount of fees regardless of the amount transferred and adapts it only from time to time. Querying this API frequently allows us to get an overview of the changes for all currencies. Additionally, the response includes the expected exchange rate and the deposit limits, which give us more insights into the service. Nevertheless, we won't need these values for the analysis process and therefore won't save them.

```
{
    "rate": "0.06057227",
    "limit": 9.74528106,
    "pair": "ETH_BTC",
    "maxLimit": 9.74528106,
    "min": 0.00196114,
    "minerFee": 0.00006
}
```


Figure 6: "Market Info" JSON Response Excerpt (*shapeshift.io/marketinfo/)*

The second useful API call is the "Recent Transaction List" request. This call returns a maximum number of 50 exchanges most recently executed by the service. For every trade, the incoming and outgoing currency, a timestamp, the deposit amount and an intern transaction ID (which changes if the request is sent once again) are revealed. The information is general, and as no addresses or hashes are included we cannot connect it to data on the blockchains. Furthermore, we cannot calculate the exact exchange rate for a given trade, because the withdrawal amount is missing. Nevertheless, this API call provides us with a starting point for scraping a large number of exchanges with partial data.


```
{
    "curIn": "ETH",
    "curOut": "BTC",
    "timestamp": 1521914986.889,
    "amount": 0.01274087,
    "txid": 414815
}
```


Figure 7: "Recent Transaction List" JSON Response Excerpt (*shapeshift.io/recenttx/[max]*)

The last API request ("Status of deposit to address") returns the most detailed data. It is possible to get status information about an exchange by passing Shapeshifts deposit address to it. If the transmitted address is a real Shapeshift deposit address, the JSON response always consists at least of the submitted deposit address and the status the exchange is currently in. The status can be either "no_deposit" (waiting for deposit), "received" (deposit was received, but the exchange is still processed), "complete" or "failed" (containing a failure message). In case an exchange is completed more information is attached to the response. This information includes the currency symbols and the amount of the incoming and the outgoing transactions, which makes it possible to calculate the exact exchange rate for a given transfer. Furthermore, the withdrawal address, which is the address the customer receives the exchanged money to, and the transaction hash of the withdrawal are returned. This data makes it possible to link two corresponding transactions on two different blockchains and identify them as a cross-blockchain transaction.

```
{
    "status": "complete",
    "address": "0x5507f17dbcdf556cb38cabc6dde4005b95038a0e",
    "withdraw": "16teePmRa9fNd6yhGmUPb2H2phznqtVpxg",
    "incomingCoin": 1,
    "incomingType": "ETH",
    "outgoingCoin": "0.05991562",
    "outgoingType": "BTC",
    "transaction": "e830db25b7202666e1d722b5700445427b43693810a7a8625cf3f82c723eb6da",
    "transactionURL": "https://blockchain.info/tx/e830db25b7202666e1d722b5700445427b43693810a7a8625cf3f82c723eb6da"
}
```


Figure 8: "Status of deposit to address" JSON Response (*shapeshift.io/txStat/[address]*)

Others

All other OTC services do not offer any REST API. Customers can only track their exchanges via the website. On Blocktrades.us, this can be done by creating an account, executing transfers through this and later retrieving the history (Blocktrades.us, 2018). On Evercoin (Evercoin, 2018) and Coinswitch (Coinswitch, 2018) customers can track their exchanges by saving the intern transaction ID after creating a new exchange. Then they can retrieve the exchange details by passing this ID to a search on the website.

Conclusion

The analysis leads to the conclusion that most instant cryptocurrency exchanges avoid revealing any information about their trades. Only Shapeshift offers few API calls that make it possible to obtain a set of transactions executed on this platform. Therefore, in the following, we will concentrate on retrieving data from this service.

3.2.2 Algorithm for Scraping Data from Shapeshift

Using the three previously presented Shapeshift API calls, it is possible to scrape an entire set of exchange data continuously. Such a scraping process consists of three main steps:

1. **Retrieving general information of the last 50 exchanges (Figure 9).** First, we must the request 50 most recent exchanges and save them continuously. The goal is to construct an array of exchanges which are sorted by timestamp. The timestamp of each entry represents the time of this particular exchange. The time is given in Greenwich Mean Time (GMT). The timestamp of the last transfer lies approximately three minutes before the time of acquisition of the dataset. The time range between the first and last exchange reaches from four to ten minutes depending on the current utilization of the service. This observation leads to the assumption that we should send a new request every at most four minutes to guarantee a complete data set. Nevertheless, requesting data every 30 seconds shows that in a new set of exchanges, which also contains exchanges that were in the previous set, new entries are added in between the already known entries. This phenomenon can be explained by the assumption that Shapeshift handles exchanges involved in different blockchains separately and confirms each group in

different time periods. Groups of exchanges with the same incoming currency are therefore added to the set of recent transactions lately. The example in Figure 10 shows the excerpt of two API responses. The right one was requested 30 seconds after the left one. The new exchanges in the second response are marked and show that transfers are added in groups having the same deposit currency. In the example, exchanges involving Bitcoin Cash (BCH) and ZCash (ZEC) are added delayed. Therefore, responses must be retrieved and aggregated continuously in a short period (in the final implementation every 30 seconds) to make sure no exchanges are missed out. Nevertheless, as we don't know in which time intervals Shapeshift updates recent exchanges of each currency, some exchanges might never be included in the response. Thus, a complete data set cannot be guaranteed. For every newly retrieved response, we add yet unknown entries to the previously retrieved ones. Then, we save all exchanges having a smaller timestamp than the smallest timestamp of the last retrieved set, as no future entry will precede it.

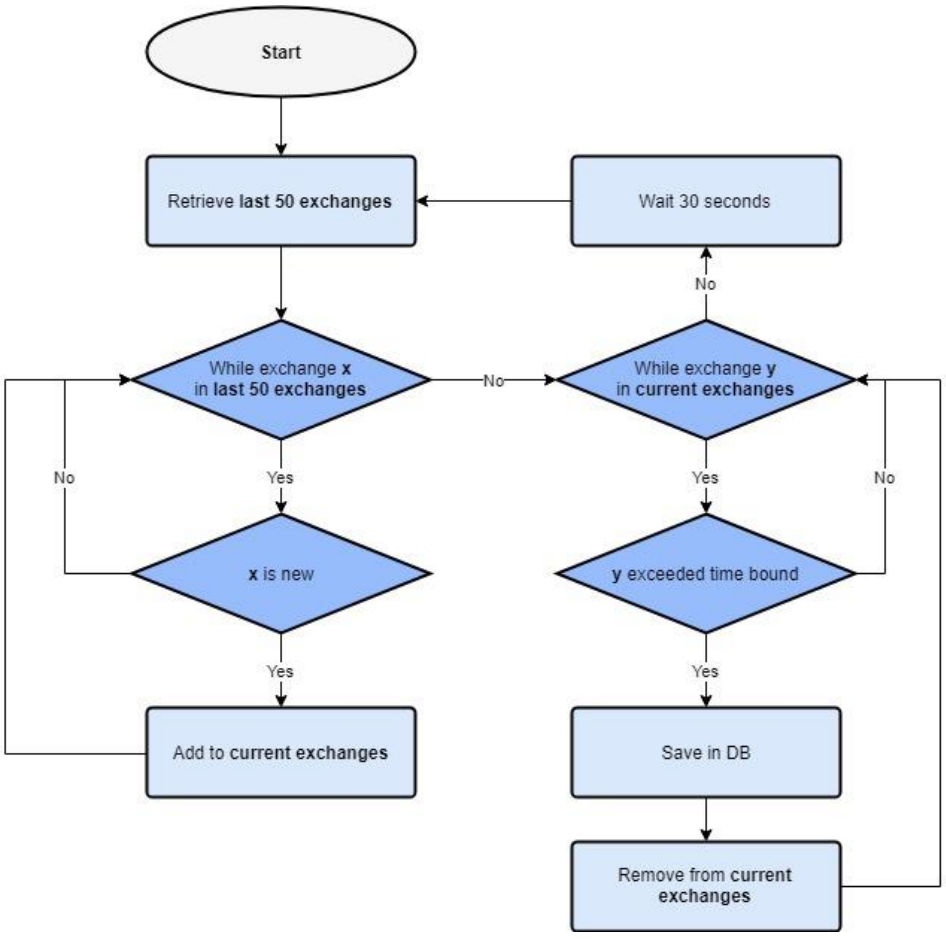
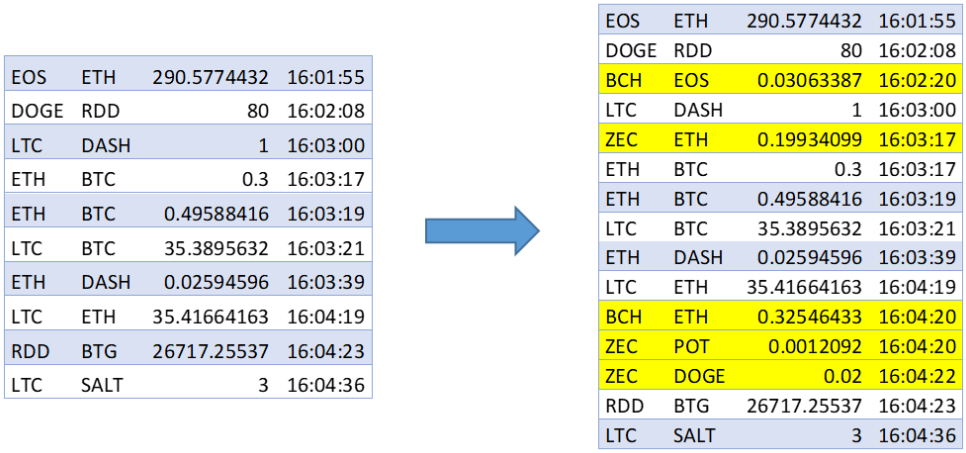


Figure 9: Retrieving Shapeshift Exchanges - Process Flow



EOS	ETH	290.5774432	16:01:55
DOGE	RDD	80	16:02:08
LTC	DASH	1	16:03:00
ETH	BTC	0.3	16:03:17
ETH	BTC	0.49588416	16:03:19
LTC	BTC	35.3895632	16:03:21
ETH	DASH	0.02594596	16:03:39
LTC	ETH	35.41664163	16:04:19
RDD	BTG	26717.25537	16:04:23
LTC	SALT	3	16:04:36

EOS	ETH	290.5774432	16:01:55
DOGE	RDD	80	16:02:08
BCH	EOS	0.03063387	16:02:20
LTC	DASH	1	16:03:00
ZEC	ETH	0.19934099	16:03:17
ETH	BTC	0.3	16:03:17
ETH	BTC	0.49588416	16:03:19
LTC	BTC	35.3895632	16:03:21
ETH	DASH	0.02594596	16:03:39
LTC	ETH	35.41664163	16:04:19
BCH	ETH	0.32546433	16:04:20
ZEC	POT	0.0012092	16:04:20
ZEC	DOGE	0.02	16:04:22
RDD	BTG	26717.25537	16:04:23
LTC	SALT	3	16:04:36

Figure 10: Comparison of two Shapeshift Responses

2. **Retrieving additional data.** In the next step, we want to enhance each exchange with more data. For this purpose, the Shapeshift "miner fee" for every currency must be retrieved in short time intervals from the "Market Info" API and saved with the corresponding exchanges. Also, the current rates in USD for every currency should be requested from a suitable extern provider and attached to each entry, to check if the real rate is comparable to an external rate. If both are similar, it should be possible to estimate the withdrawal amount from the deposit amount.

3. **Finding transactions on the blockchains (Figure 11).** In the last step, we create a connection between the scraped data and data on the blockchains. This process is repeatedly run after a fixed amount of time (in the final version of the scraper every 30 minutes) for exchanges, which were still not analyzed or not found. We can divide the process into three parts.

   First, we want to find the transaction the customer did on the blockchain of the deposit currency. For this, we get the block with the current block number of the corresponding blockchain, as the exchange was executed no longer than few minutes before retrieving it. We then iterate over the transactions in this block. The amounts given in the JSON response are exact. Therefore, we can compare if the value of a deposit is the same as the value of one of the outputs of a transaction. It should be beard in mind that inputs are the money sent by the customer, but as also fees must be paid to the network, only the output can correspond to the deposit amount. If we can't find the corresponding amount the next block is retrieved and analyzed. We execute this process until we find a matching transaction or a limit is surpassed.

   In case of a match a request with the address to which the found output was sent, is forwarded to the "Status of deposit to address" API. If this is a real Shapeshift exchange, the API should return the detailed information about this transfer, with which we can enlarge the data of the entry. In case the API doesn't return exchange details, the found transaction is not the right one, and we must continue searching. It can also happen that exchange details are returned, but don't belong to the found

transaction. There are two reasons this can happen. First, the currency of the response in the first step does not fit the one returned from this API call. We encounter this problem with tokens implemented on Ethereum. In the first response, the currency is stated as Ether, because Ether must be sent to Shapeshift to cover the fees for executing the Smart Contract which transfers the tokens, whereas the second response returns the currency and the amount of the token. We can solve this problem by checking if the deposit amount and symbol are still the same. The second reason is if a customer uses a deposit address multiple times. In that case, the API only returns the information about the last of these exchanges. So, if the customer executes various trades with the same deposit address in a short time interval which is smaller than the time interval of this process, we cannot find additional information for all exchanges anymore. Nevertheless, the possibility for this to happen is very low and we can prevent adding wrong information by checking if the incoming amount is the same.

Finally, we must find missing data for the withdrawal transaction, like the transaction fee and the block number. As we already get the transaction hash of this transaction from the API response in the previous step, we can easily query all details by searching by this hash. It can also occur that no transaction hash is shown in the answer. This happens if an exchange was canceled, due to a wrong deposit amount, the late arrival of the deposit or other reasons stated in a message included in the response. In this case, the exchange wasn't successful and doesn't have to be tracked.

With the outcome of this process, a significant record of exchanges with detailed data can be generated, which helps us to analyze the transfer processes of Shapeshift in detail and is useful for the evaluation of the outcome of the implemented tool.

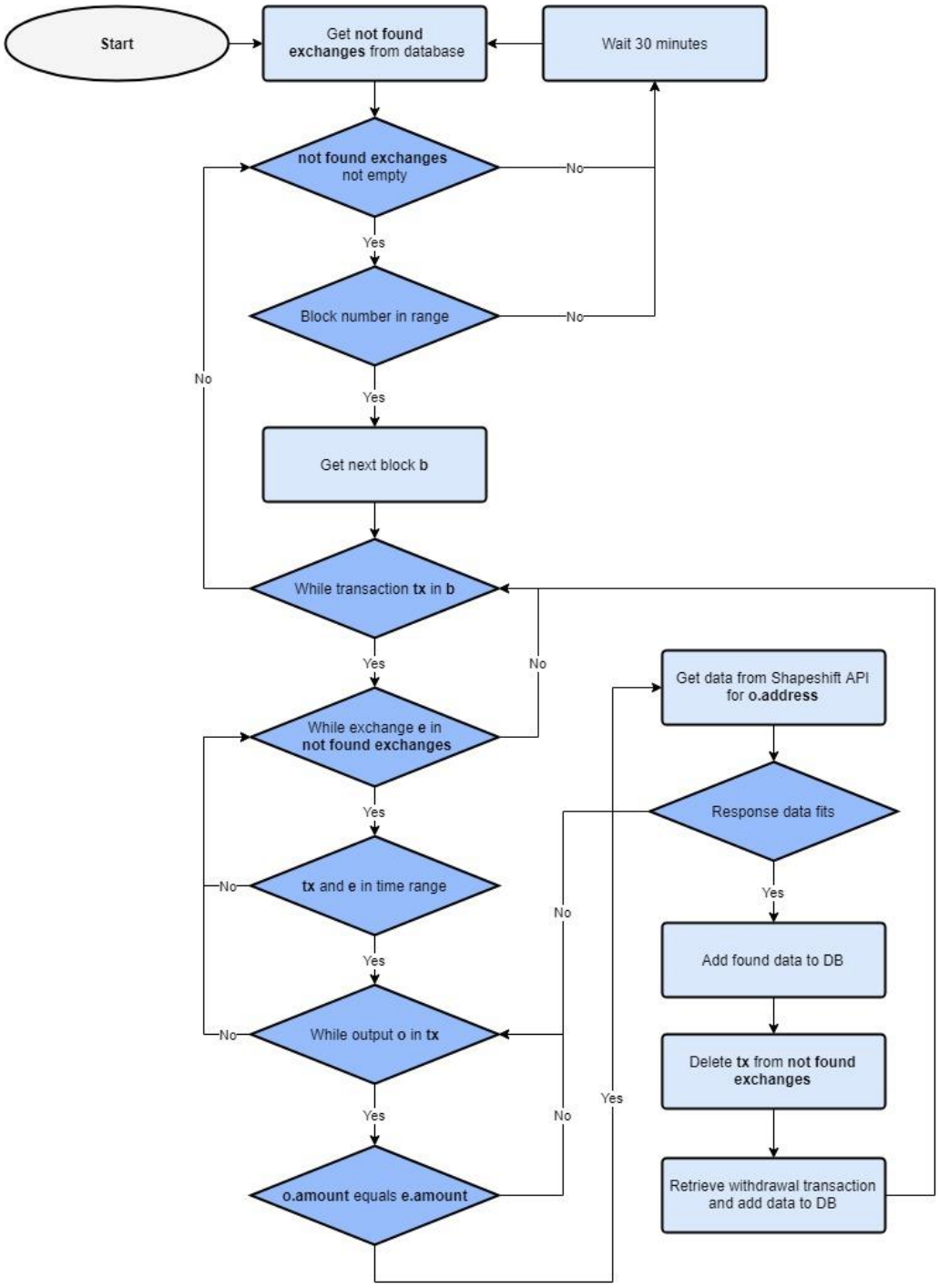


Figure 11: Finding Blockchain Data - Process Flow

3.3 Analysis and Findings

We can assume that Shapeshift uses standardized and automatized processes to handle the high number of requests. In the first place, this includes receiving and sending of money on all blockchains networks, which are involved in the service. Additionally, the appropriate distribution of funds on different addresses must be handled, to guarantee

liquidity and be able to operate customer requests in short time. In this chapter, we want to analyze this complex structure of transaction flows.

3.3.1 Analysis of Scraped Data

In the following, we take a closer look at the scraped data to find patterns. For this two sets of scraped data are analyzed. The first one includes exchanges scraped on the 14. of December 2017 in a time range of 5 hours. It has 1,742 entries in total, of which 318 are exchanges from Bitcoin to Ether and vice versa. This data was scraped with a first version of the scraping algorithm and didn't track all exchanges provided by the Shapeshift API. Nevertheless, it is valuable for analysis. The second set was scraped over a time range of approximately two weeks between the 1. and the 14. of February 2018. It consists of over 105,000 entries, of which over 21,000 are exchanges between Bitcoin and Ether. The analysis will take account of every attribute of the data sets.

Currencies

The sets show that most of the retrieved exchanges transfer Ether to Bitcoin. For the first set this concerns all trades except two, and for the second set, it is almost 80%. Reasons for this distribution will be analyzed and explained after the implementation of the recognition tool.

Amount

The range of the incoming exchange amount in both data sets is similar. The highest exchanges are 0.83 BTC and 8.71 ETH. This result also matches with the limits of the service exchange set by the service.

Transaction Fee

As Shapeshift defines the exchange fees before the exchange, it could maximize its profit by setting a small transaction fee when sending the withdrawal. Nevertheless, the scraped data shows that the fees paid by Shapeshift are in a normal range. For Bitcoin, e.g., they were between 0.0007 and 0.0017 BTC in the data set from December. This leads to the assumption that transaction fees are calculated considering the current average transaction fees, to guarantee that a transaction is confirmed quickly.

Exchange Fee

Shapeshifts fee is adapted over time to be in line with the development of the transaction fees on the blockchain networks. The scraped data shows this precisely. In December the fee was 0.00175 BTC (ca. 29$) for exchanges to Bitcoin and 0.01 ETH (ca. 6$) for

exchanges to Ether. In the first week of February, the fees lied between 0.00065 BTC – 0.0011 BTC (5-10$) for exchanges to Bitcoin and 0.002 – 0.003 ETH (1 – 3$) for exchanges to Ether. The fee was much higher in December due to the higher network fees caused by the high demand for cryptocurrencies at the end of the year 2017.

Another finding regarding the exchange fee is that Shapeshift tries to lower it by doing multiple withdrawal transactions within one transaction. Therefore, the exchange fee for one exchange can be even lower than the fee paid by Shapeshift for executing the withdrawal transaction. This is possible as the Bitcoin protocol allows to have multiple outputs to different addresses in one transaction. This method can't be applied for Ethereum withdrawals as transactions only have one output.

Address

The found data shows that all deposit addresses for receiving Bitcoin start with a "3". Such addresses are classified as P2SH (Pay to script hash) addresses. Behind this hash, a script is implemented, which secures the spending of deposits. It can, e.g., be a multi-signature script, which requires the signing of a transaction by multiple people or a password (Tschorsch, & Scheuermann, 2016). Apart from that, no patterns or multiple occurrences of a single address can be recognized. This is due to the fact, that Shapeshift generates a new address for each deposit, except the user requests to use an address multiple times. The found addresses will be useful for tracking money flows and recognizing the process structure. We describe this work in the next chapter.

Times/ Time Differences

For the scraping process, we use the API of Blockchain.info to get Bitcoin blockchain data, as it has the advantage that apart from the block confirmation time, it also returns the transaction time. As explained before, this is important for the analysis, because these two times can significantly differ from each other. For Ethereum, we use the Infura service, which returns the same output as a request to an own full node. Therefore, it only contains the confirmation time of the block. As the average time for adding a new block to the public ledger is short, mostly taking just a few seconds, it can be assumed that the difference between receiving the transaction and confirming the block is so small that no attention must be paid to it in this analysis.

The scraped data reveals the order a transaction is processed. The process flow is presented in Figure 12. The scraped data shows that Shapeshifts exchange timestamp mostly lies only a few seconds after the time the deposit transaction was received in the network. Outliers, where the transaction time lies just a few seconds after the exchange time, can be explained by the fact that Blockchain.info and Shapeshift don't receive a transaction at the same time from the network. Due to this, the recognition of the same transaction can slightly differ in time. Nevertheless, we can conclude that Shapeshift waits for a customer transaction to appear in the network to set the timestamp which can be retrieved from the API later. Then, Shapeshift waits that the deposit is confirmed within

a block and then send the withdrawal transaction. The time the withdrawal is received in the network is in average approximately two minutes after the time the deposit was confirmed. In this time the rate for the exchange is recalculated again, as some time has passed. Therefore, the outgoing amount slightly differs from the estimated amount at the beginning of the process. The whole exchange ends when the withdrawal transaction is confirmed as well. The average time for the entire trade from sending the deposit transaction to the confirmation of the withdrawal transaction takes around 10 minutes. The average exchange time shown on the main Shapeshift page is mostly lower, as we only consider exchanges involving Ethereum and Bitcoin. Most of the other cryptocurrency networks proceed transactions faster than the Bitcoin network, which also leads to a faster trade.

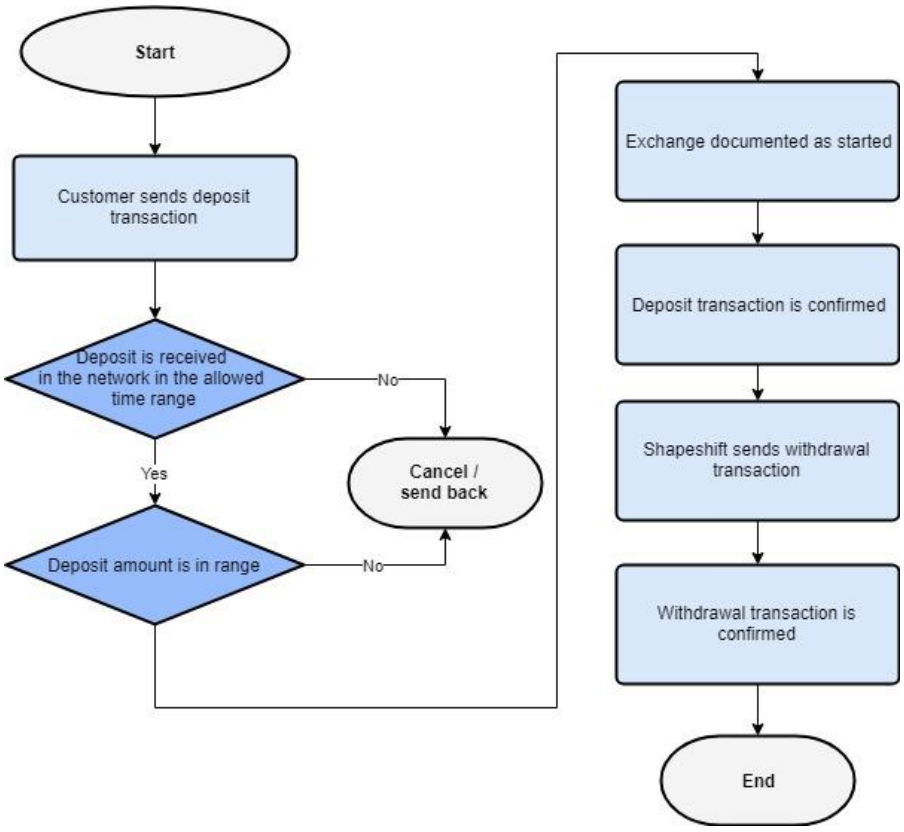


Figure 12: Shapeshift Exchange - Process Flow

Exchange Rate

As we simultaneously scraped the Shapeshift fee and the actual dollar values for all cryptocurrencies, we can check by which percentage a calculated expected outcome based on these parameters would differ from the real outcome. For this we use following calculation:

$$\text{expected amount} = (deposit\ amount * exchange\ rate) - exchange\ fee$$

$$exchange\ rate = \frac{value\ of\ deposit\ currency\ (in\ USD)}{value\ of\ withdrawal\ currency\ (in\ USD)}$$

On average the expected amount is 9% higher than the real outcome in the data set retrieved in December and only 0.5% higher at the beginning of February. We can see that there are some outliers, mostly when the exchanged amount is very small. The smaller the amount is, the bigger is the difference, as the exchange fee can nearly be as high as the real output amount. Without an exact rate prediction, the approximation is impossible. To see how our estimation would be without those small values we calculate the average difference again after filtering those exchanges. In December the exchange fee was ordinary high. Transfers to Bitcoin were charged with approximately 30$ of Shapeshift fee. Filtering exchanges with a lower output than 35$ we get an average difference from the real value of -0,6%. At the beginning of February, the demanded fee was up to almost 10$. Filtering all exchanges below this amount shows that the expected amount lies 0,8% under the real output. This result indicates that a good approximation can be reached with an externally retrieved exchange rate. Only exchange amounts that are nearly as big as the exchange fee can't be surely found without the exact rate used for the exchange.

Conclusion

The only parameters from the raw block data that are useful for recognizing if two transactions on different blockchain are connected to each other are the timestamps, which can be matched to each other by an expected time difference, and the amounts involved, which can be matched using an external exchange rate.

The next step is to analyze the transaction processes of Shapeshift manually with the help of the scraped addresses, using blockchain explorers with which transactions can be followed step by step. This process is performed separately for Bitcoin and Ethereum. Within this analysis, multiple addresses connected to Shapeshift were found and categorized.

3.3.2 Analysis of Ethereum Transactions

Main Address

Etherscan.io is an explorer for the Ethereum blockchain, with which it is possible to search data by address or transaction hash. It also offers the feature to label known addresses of companies and services. The main address of Shapeshift is also marked and therefore easy to find.

NAME	ADDRESS
Shapeshifts Main Address	0x70faa28a6b8d6829a4b1e649d26ec9a2a39ba413

This address was created in October 2016 is used as a central point for distributing Ethers, therefore receiving money from different sources and sending it to different recipients.

Until February 2018 this address was used for more than 1.4 million of transactions. We will analyze incoming and outgoing transactions separately.

Incoming Transactions

Figure 13: Incoming Shapeshift Transaction for Ether

We can observe from the scraped data that every deposit address forwards the whole received amount of money to the main Shapeshift address after a certain time. Mostly this is done shortly after receiving the deposit, but outliers also could be detected, in which a deposit was sent more than a month after receiving it. As the customer doesn't have to specify the address he will send the money from, Shapeshift must generate new addresses for every user, to be able to recognize which deposit belongs to which exchange. Sometimes the customer uses the same address for multiple exchanges. In that case, the deposits are forwarded to the main address either stepwise or in one output transaction.

Additionally, other incoming transaction to the main address could be recognized which originate from different sources. For instance, inputs from different trading platform services, whose address were also labeled by Etherscan.io. Four of these services could be detected: Bitfinex, Poloniex, Bittrex, and Binance. The labeled addresses are shown in the table below.

OWNER	ADDRESS
Bitfinex	0x876EabF441B2EE5B5b0554Fd502a8E0600950cFa
Poloniex	0x32Be343B94f860124dC4fEe278FDCBD38C102D88
Bittrex	0xFBb1b73C4f0BDa4f67dcA266ce6Ef42f520fBB98
Binance	0x3f5CE5FBFe3E9af3971dD833D26bA9b5C936f0bE

Outgoing Transactions

The scraped data reveals that all withdrawal transactions are send from exactly four different addresses:

NAME	ADDRESS
Withdrawal Address 1	0xd3273eba07248020bf98a8b560ec1576a612102f
Withdrawal Address 2	0x3b0bc51ab9de1e5b7b6e34e5b960285805c41736
Withdrawal Address 3	0xeed16856d551569d134530ee3967ec79995e2051
Withdrawal Address 4	0x563b377a956c80d77a7c613a9343699ad6123911

These addresses are not labeled on Etherscan.io. Each of them is used since June 2017, was involved in more than 250,000 transactions (until February 2018) and mainly has outputs. The only input is a periodic payment from Shapeshifts main address, which guarantees that these addresses can continuously send money to the customers. The input is always a payment of around 400 Ether and is sent roughly every day. As the total number of transactions is comparable and all addresses continuously send money, we can assume that Shapeshift distributes all withdrawal transactions on these four addresses equally. The scraped data also reveals that the address used for a particular exchange output doesn't depend on the incoming currency of the exchange.

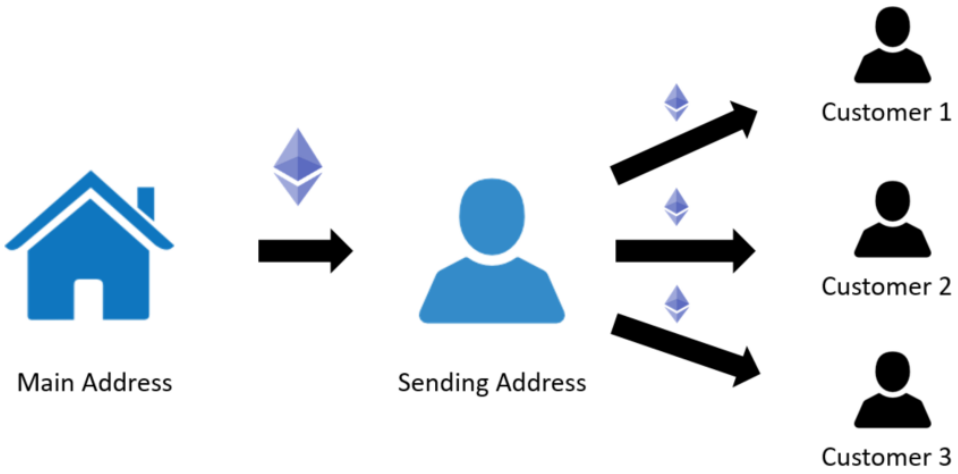


Figure 14: Outgoing Shapeshift Transaction for Ether

Exchanges, which involve tokens based on Ethereum, are handled by Shapeshift over special Ethereum addresses. Each of these addresses is responsible for transactions of a certain token. An amount of around 2 Ether is send to each of these addresses frequently. This money is used for paying the fees required for calling the smart contract, which executes a token transfer. Just as for the four forwarding addresses stated before, this amount is always sent before the address runs out of money. Following token addresses could be identified (for some tokens there are multiple addresses):

TOKEN	ADDRESS
OmiseGo (OMG)	0xA7170FBEBace7F13D0BB82FA04eA6A36c0A576b7 0x5E44c3E467a49C9Ca0296a9F130fc433041aAa28
EOS (EOS)	0x59f1e1C1EFe5D1350287862Ba53A61f1dC3B78FD 0x692DA4782d996DAC7D66B5822f3c504f67dA8493
Golem (GNT)	0x7fe2b88f2e4858de375832fbf54ac7cf1a78ca51
Status Network (SNT)	0x2E46956565CEbDcBbB39EcD22aF02E1916a2FE37
Aragon (ANT):	0xebfea9697bc8fde56b142c57de59136481785fa1
Basic Attention (BAT):	0x73295d3c0ca46113ca226222c81c79adabf9f391
Funfair (FUN)	0xdf04eaf5fe642ab9fce3a9bb4957361f514bc657
District0x (DNT)	0x412ce78c6cb4c227e1d1522ba484b4cc8c051b13
Salt Lending (SALT)	0xb7Bd981cAC9f087177fE90FC4D6439d3F2782061
Matchpool (GUP):	0x54638372273d424121485eE14376EC341c0294c7

Furthermore, also outputs to trading platforms could be identified. These are realized over specific addresses as well. These addresses receive high and even amounts of Ether and forward them to the addresses of the trading platforms. Shapeshifts forwarding addresses are shown in the table below.

OWNER	ADDRESS
Bittrex	0xE9319eBA87Af7C2fc1F55ccDe9d10eA8efbd592d
Bitfinex	0xDa1E5D4Cc9873963f788562354b55A772253b92f
Poloniex	0xe8ed915E208B28c617d20F3F8Ca8e11455933aDf
Binance	0xb36eFd48c9912Bd9fd58b67b65f7438F6364a256

3.3.3 Analysis of Bitcoin Transactions

In contrast to Ethereum, Bitcoin transactions can contain multiple inputs and multiple outputs. As explained in the second chapter a received input can't be divided and must be spent entirely. Nevertheless, inputs mostly don't correspond exactly to the amount the sender wants to transfer. That is why transactions don't only contain outputs which belong to the desired recipients, but also other addresses belonging to the sender, to which the rest is sent back. These addresses are called change addresses. So, apart from identifying which transactions are related to Shapeshift, it will also be required to understand which inputs and which outputs of this transaction belong to which party.

Incoming Transactions

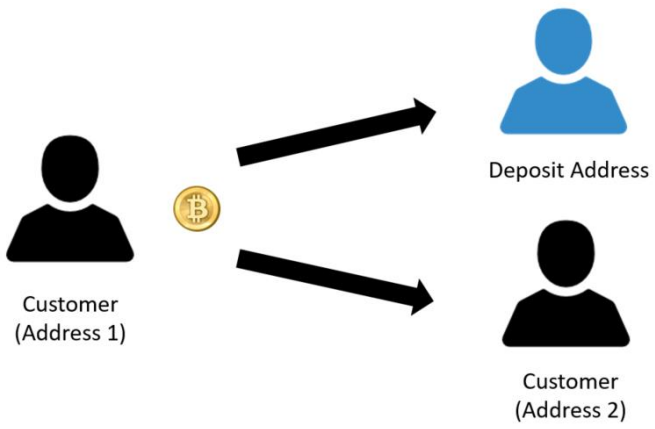


Figure 15: Incoming Shapeshift Transaction for Bitcoin

As stated before customer deposits are exclusively sent to P2SH addresses. Furthermore, the analysis of the Shapeshift transactions showed, that this kind of addresses are only used for receiving money from the customer. Therefore, if a transaction can be identified as related to Shapeshift, it can also be easily categorized as a deposit. The service uses the deposited money in an outgoing transaction. In case a deposit address is used multiple times, each input is spent separately in different output transactions. Shapeshift mostly forwards the deposit within a day. However, in some cases, it can also take more time.

For all other incoming transactions, Shapeshift owns a separate Bitcoin address. Here, it receives more significant amounts of money and then uses it for sending it to the customers. Inputs include payments from trading platforms. Shapeshift uses this address since April 2017 and has received and forwarded over 107,000 BTC since then (BicoinInfoCharts, 2018).

NAME	ADDRESS
Shapeshifts Main Deposit Address	1NSc6zAdG2NGbjPLQwAjAuqjHSoq5KECT7

With the help of the services bitcoinwhoswho.com and blocktrail.com, it was possible to identify incoming amounts from the same four trading platforms, as used for exchanging Ether. These are their Bitcoin address:

OWNER	ADDRESS
Bittrex	1N52wHoVR79PMDishab2XmRHsbekCdGquK 14cQRmViAzVKa277gZznByGZtnrVPQc8Lr
Bitfinex	1Kr6QSydW9bFQG1mXiPNNu6WpJGmUa9i1g
Poloniex	12cgpFdJViXbwHbhrA3TuW1EGnL25Zqc3P
Binance	1NDyJtNTjmwk5xPNhjgAMu4HDHigtobu1s

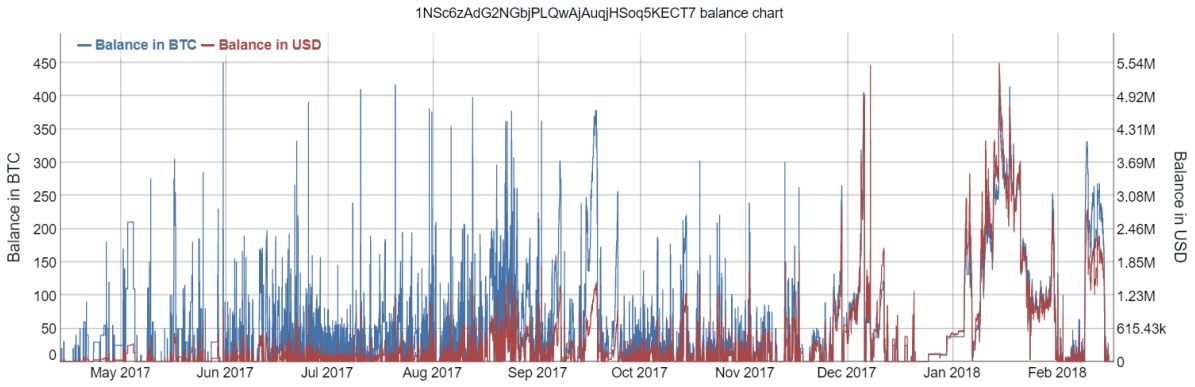


Figure 16: Shapeshifts main deposit address – Balance Overview (BitInfoCharts, 2018)

Outgoing Transactions

In comparison to the processes on the Ethereum network, Shapeshift has no particular withdrawal addresses that handle all outgoing transactions for Bitcoin. A customer withdrawal is derived either directly from an incoming or another outgoing transaction.

There are two ways Shapeshift handles the inputs of such a withdrawal transaction. Either many small UTXOs, especially those received from the customer as a deposit, are combined to satisfy a high enough amount for realizing an output. In case a Shapeshift address holds an UTXO with a high number of Bitcoins no UTXOs must be combined. To accomplish such a high-value UTXO, Shapeshift either sends a high-value output from its main deposit address or uses temporary addresses in which small value UTXO are

inputted, and high-value UTXO are created. An example is the address listed below which was used for summing up UTXO within a day.

NAME	ADDRESS
Shapeshifts Temporary Address	1B6MUdDVNZU5tEWoLLcqVVk6GU2GgUiHq6

We can see that a transaction can either contain just one output being a withdrawal or have multiple of such outputs handling multiple exchanges at the same time. Mostly these transactions contain one output address, which is a change address and therefore belongs to Shapeshift. This output is used again as an UTXO in another outgoing transaction.

The payments are therefore not handled by just one main address, but by a chain of single addresses. The difference in the underlying protocol can explain the difference in processing between Ethereum and Bitcoin. As described in chapter 2, addresses are seen as separate accounts in Ethereum. Thus, it is not possible to send one transaction from multiple addresses. That is why Shapeshift must bring together all deposits, which mostly are small amounts of Ether. Then it is possible to send payments of any height from this address. For Bitcoin transaction, multiple UTXO of different addresses can be added together and so a main address is not needed. Moreover, the sending to a main address would cause additional transaction fees, which are much higher in comparison to Ethereum transaction fees.

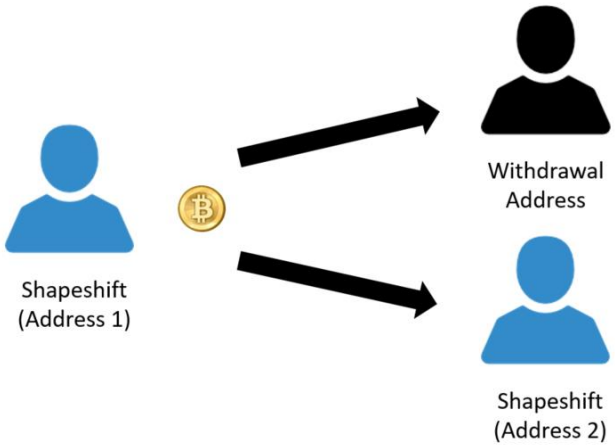


Figure 17: Outgoing Shapeshift Transaction for Bitcoin

External Deposit Addresses

In comparison to Ethereum, there is no blockchain explorer, which labels a Bitcoin address as related to Shapeshift. As stated before Shapeshift doesn't have a main Bitcoin address. Nevertheless, addresses, continuously receiving a high number of Bitcoins from Shapeshift related addresses, could be identified by following the transaction flow which originates from the scraped addresses. Almost all customer deposits end up in one of these addresses, either directly or through a series of outgoing transaction.

Firstly, we could uncover four addresses that belong to trading platforms and to which Shapeshift sends deposits continuously. These addresses are exclusively for Shapeshift and only get inputs from it. The four trading platforms are the same as identified for the incoming transactions.

OWNER	ADDRESS	USER SINCE	AMOUNT RECEIVED
Bittrex	1NoHmhqw9oTh7nNKsa5Dprjt3dva3kF1ZG	January 2015	>27,000 BTC
Poloniex	1BvTQTP5PJVCEz7dCU2YxgMskMxxikSruM	March 2015	>99,000 BTC
Bitfinex	1LASN6ra8dwR2EjAfCPcghXDxtME7a89Hk	August 2016	>58,000 BTC
Binance	17NqGW6HY3f2LY7wFkEDn9yXpq8LWMRMEQ	February 2018	>1,700 BTC

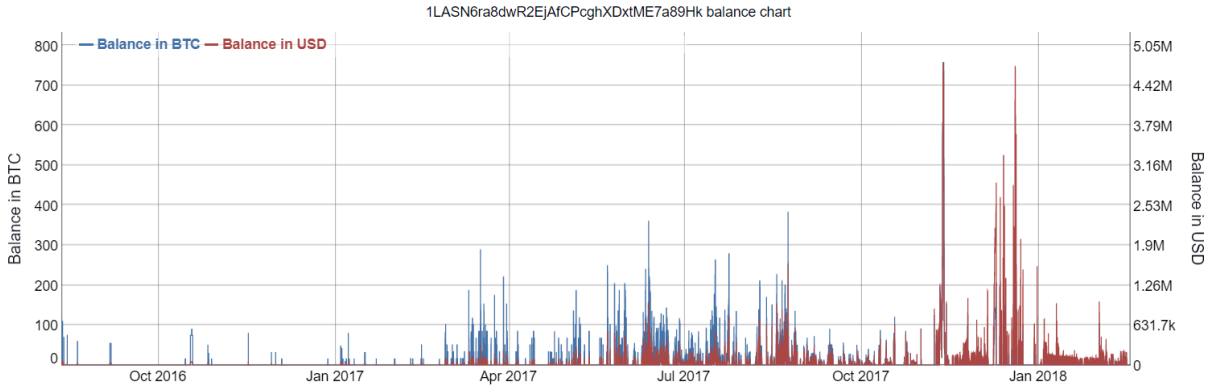


Figure 18: Shapeshift Deposit Address for Bitfinex – Balance Overview (BitInfoCharts, 2018)

Furthermore, we found a deposit address, which mainly has inputs and holds a high number of Bitcoins (over 2,600 BTC in February 2018). These characteristics lead to the assumption that this is one of the Bitcoin addresses on which Shapeshift stores money not used in the exchange process anymore. Therefore, this could also be one of Shapeshifts cold wallet addresses. This address has a small amount compared with the cold wallets of the trading platforms Bitfinex and Bittrex, which hold more than 150,000 BTC (March 2017) and are two of the wealthiest Bitcoin addresses (Richest Bitcoin Addresses, 2018). Bitfinex states that their cold wallet contains 99.5% of all user funds (Bitfinex Cold Wallet, 2018). The wallet structure of Shapeshift must be different, as user funds are processed within a couple of minutes and not kept over a long term. Nevertheless, we can assume that the service holds their profits on offline storages as well. Except for this address, no transactions to other cold wallets could be identified on neither of both presented networks. Shapeshift could also realize this, e.g., by transferring the money intended for storing directly from its trading platform accounts to its cold wallet accounts, what would have no connection to the scraped data and so be untraceable for us.

OWNER	ADDRESS
Storage Address	3K9Xd9kPskEcJk9YyZk1cbHr2jthrcN79B

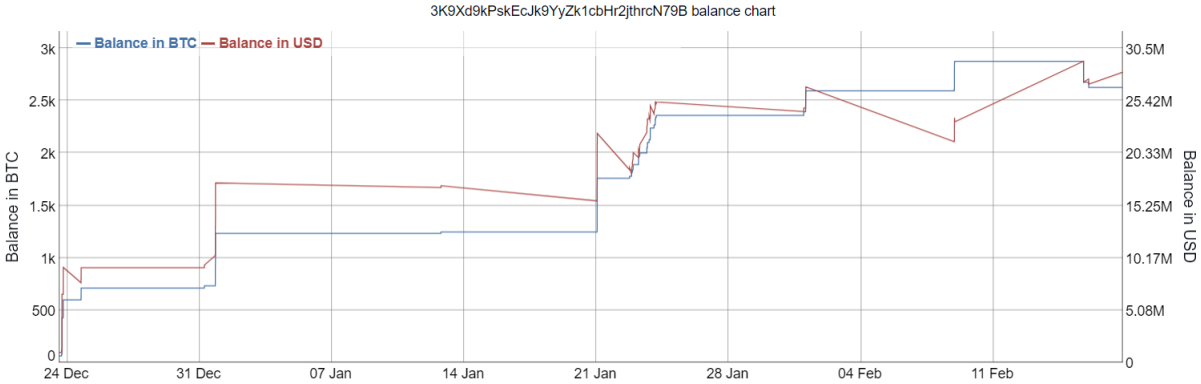


Figure 19: Shapeshift Address for Storage – Balance Overview (BitInfoCharts, 2018)

Finally, we could identify addresses, which first receive multiple deposits from Shapeshift during a short period – similarly to the temporary addresses. Then they send a high amount to unknown addresses, which can't be identified as Shapeshift addresses. The amounts are not forwarded anymore and therefore end the chain of outgoing transactions.

OWNER	ADDRESS
Unknown 1	1NE6snFBUQD2aExH8KZdzEbDiNCyANjfVg
Unknown 2	1GJkx984EHyR5dCPvVisE9Y7p18MKa1ixs

Conclusion

Shapeshift distributes its money by sending it through multiple flows of transactions. Such a flow originates from a deposit transaction or a transaction sent by Shapeshifts main deposit address. Then the money is forwarded over one or multiple outgoing transactions. Finally, the transaction flow ends in one of the external deposit addresses. Flows are often merged, e.g., by combining various deposit UTXO.

Note

All addresses presented were lastly checked at the end of April. Transaction done afterward are therefore not included in this analysis.

For both, the Bitcoin and Ethereum transaction processes, single transactions were found which couldn't be assigned to any of the address categories presented in this chapter. Nevertheless, these are mostly transactions with a small value. All transactions including high-value transfers and those related to customer exchanges could be identified here. Therefore, we can use these findings to construct heuristics for recognizing cross-blockchain transactions.

3.4 Heuristics for Cross-Blockchain Transaction Recognition

In this chapter, we want to construct heuristics with which it is possible to recognize cross-chain transactions initiated by the Shapeshift service using only data publicly available on the blockchains or other public services. From the previous analysis, three parameters could be identified as useful for such a recognition algorithm.

3.4.1 Time Comparison

The first step for matching transactions from two blockchains to each other is to check if they were created in a specific time range and so can be candidates for building a possible exchange. To minimize the number of matching transactions, the shortest time difference between these two transactions must be calculated. As previously explained, this is the range between the block confirmation time of the deposit transaction and the transaction time of the withdrawal transaction. The average duration is 2 minutes, but an appropriate interval should be set to find as many correct pairs as possible. For the lower range, we can just set the difference to 0, as Shapeshift can send the withdrawal within a few seconds after receiving the deposit. We found exchanges, which had a long processing time (up to 45 minutes) in the scraped data. Setting such a high upper bound however would dramatically raise the number of wrongly matched pairs. The analysis of over 20,000 scraped exchanges in February shows that setting the upper bound to 15 minutes includes 99,6% of all found exchanges and therefore is an appropriate limit.

3.4.2 Exchange Rate

The next step is to check if the transaction pairs also match regarding their values. We saw that the real exchange rate could be approximated adequately with an external exchange rate, except for tiny amounts. Therefore, we must retrieve the exchange rate for the two currencies and check if any of the output values of the transactions match using this rate, considering the exchange fees as well. A match is found when the withdrawal amount is in a specific range around the calculated expected outcome. We must select this range in such a way that we minimize the number of wrong pairs. Analysis of the scraped data of February showed that setting the range between -10% and +10% includes 99% of the found exchanges. The publicly available historical rates can only be retrieved in an hourly range and therefore won't be as precise as the rate that is continuously actualized by the scraper in a shorter time range. Setting a higher range would guarantee to find a higher number of right pairs, but also would increase the number of false pairs noticeably.

For the calculation of the expected output Shapeshifts fee is needed. As there is no historical data for this rate, we will have to estimate it. We can do this estimation by calculating it based on the fees Shapeshift paid for the withdrawal transaction. As mentioned before, the exchange service claims a lower fee if multiple withdrawals can be handled in only one transfer. For Bitcoin transactions which involve only one or two withdrawals and for all Ethereum transactions which can only transfer one withdrawal

due to the protocol definition, the exchange fee is higher than the network fee of the transaction. The scraped data reveals that the ratio in average is +60% for Bitcoin and +80% for Ethereum transaction. For Bitcoin transactions handling more than two withdrawals, the exchange fee is significantly lower than the fee paid for doing the transfer. In average the exchange fee is 30% of the transaction fee. We use these values to estimate the Shapeshifts fee.

3.4.3 Address Recognition

Running the tool comparing only the times and values showed that the number of wrong pairs is exceptionally high, as many transactions with a similar or even the same amount are executed at almost the same time on the blockchains. Therefore, we must reduce the found set by an additional heuristic. The main problem is that all transactions from the blockchains are involved in the comparison algorithm. To get a better result, we should reduce the transactions compared exclusively to transfers related to Shapeshift. Therefore, we want to design an algorithm which recognizes deposit and withdrawal transactions for Ethereum and Bitcoin. For this purpose, we use the findings about Shapeshifts processes previously presented.

Ethereum

Since deposits, as well as withdrawals, are connected to the labeled main address of Shapeshift over one additional hub, the classification of a given transaction to one of this two classes can be realized straightforward. Figure 20 shows the recognition process for one block.

Ethereum deposits of a customer are always forwarded after a certain amount of time to the main address of Shapeshift. That's why a transaction can be recognized as a deposit transaction to Shapeshift if there is another subsequent transaction that sends money to the main address and whose input address equals the output address of the deposit transaction, respectively the deposit address. To be able to make such a check we must retrieve an array with all transactions to the main address, which were made a certain time after the transaction under investigation. As the scraped data showed, we would have to get all transaction within the coming month to be sure to verify all deposit addresses correctly. Nevertheless, to check every transaction with all Shapeshift transactions of a whole month would noticeably reduce the performance of the tool so that a lower limit should be set. A range of 1.5 days guarantees that 99% of all deposit transactions will be found. The retrieving of the transactions of the Shapeshift main address can be done in two different ways. Either it can be retrieved using the API of Etherscan.io, which allows getting all transactions of a given address in a given time range. On the other hand, we could also retrieve all Shapeshift related transactions without using the Etherscan.io API. For this, all subsequent blocks in the range of 1.5 days should be retrieved and for all contained transactions it must be checked if the Shapeshift main address is involved as output address. Assuming the average block creation time of 15

seconds, 8,640 blocks (4 blocks/minute * 60 minutes * 24 hours * 1.5 days) have to be checked before starting the analysis.

Withdrawal transactions can be recognized in two different ways. As showed before, larger amounts of money are sent to forwarding addresses frequently, which then execute the withdrawal transactions. Since June 2017 four of such addresses are used. Therefore, for transactions done after this date, we only must check if an Ethereum transaction has one of these four addresses as an input to identify it as a Shapeshift withdrawal. Another heuristic would be to check if the input address of a transaction corresponds to an address to which the main Shapeshift address sent a larger amount of money in the past period. The analysis showed that this happens mostly within a day. Therefore, the check of a specific address should be done with all transactions of the main address which were executed at most one day before this transaction. The retrieving of the main address transactions can be done in the same two ways proposed for the recognition of a deposit. It would be required to load 5,760 blocks (4 blocks/minute * 60 minutes * 24 hours * 1 day) before the block where the transaction being checked is contained. The second method can be used without the knowledge of the addresses used by Shapeshift for paying out exchanges. Assuming that Shapeshift used the same structure for forwarding addresses before June 2017, the second heuristic can be applied for a more flexible recognition of withdrawals.

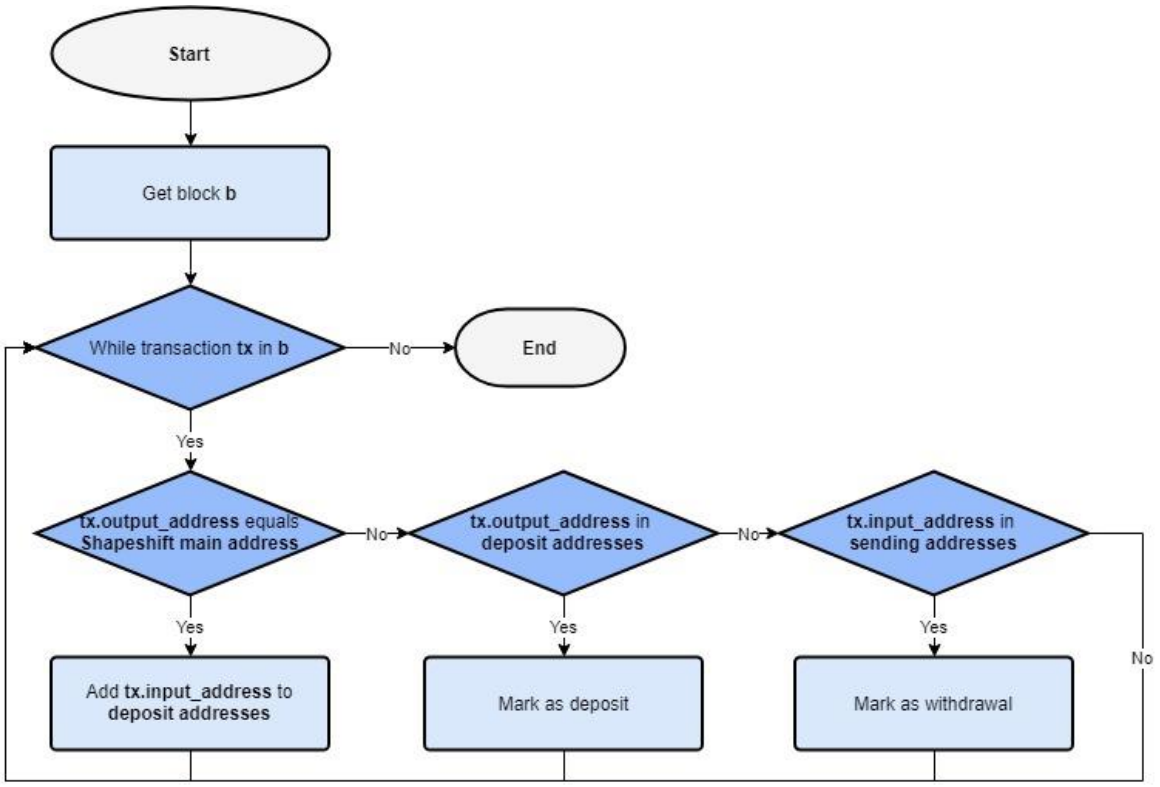


Figure 20: Address Recognition Process for Ethereum

Bitcoin

As there is no main Bitcoin address standing in the middle of deposit and withdrawal transfers, the classification must be realized by iterating over each hub of a flow of transactions, as explained in the chapter 3.3.3. We found out that such a flow starts with a deposit transaction of a customer or by a transfer from Shapeshifts main deposit address. Then, none, one or multiple outgoing transactions follow each another and so create a transaction chain. Thereby, several of these chains can come together and build a new single chain. Also, after aggregation they can be split up into multiple chains with high UTXOs, what is done in temporary addresses for merging small UTXOs. Each of these chains ends in a transaction to one of the previously identified deposit addresses to external services or cold storages. Transactions in between can't be recognized as related to Shapeshift without having the knowledge that they are contained in a Shapeshift transaction flow. So, to identify all members of a chain, we need to find each transaction step by step by starting at one of the ends of the chain. The problem by starting at the beginning of the chain is that we cannot say which P2SH transactions are related to Shapeshift without using its API. Also, assuming we could identify the starting customer deposit transaction, we would need to pass through the blocks in rising number and thus should be able to say which one of the outputs of each transaction is an address that belongs to Shapeshift and which one belongs to the customer. We can't fulfill a secure classification only having the output values as parameters. We can only identify a single Shapeshift related transaction certainly if one of the deposit addresses to the external services is involved. Therefore, these kinds of transactions can be used as a starting point for the recognition of a transaction flow. That means the recognition process starts at the end of every chain and other members of it are found by going through the blocks in decreasing number. The process follows these steps:

1. First, we create different buckets to which we assign addresses related to Shapeshift. In the beginning, we only have one filled bucket, which contains all external deposit addresses. Additionally, we have two more buckets which will be filled with the addresses involved in customer exchanges later in the process.

2. We now iterate over all Bitcoin transactions of a block and check the output addresses of each of them. If one equals the address contained in one of the buckets, we can identify the transaction as an exchange transaction.

3. Then we have to differentiate whether the transaction including the matching output is a deposit or a withdrawal:

    a. If the output address is not a P2SH address, we can label it as a withdrawal. Then, we can delete this output, as for the further process we only need the other outputs, which contain the amount and address of the withdrawal to the customer. Now that we know that this is a transaction done by Shapeshift, it is sure that all input addresses contained also belong to the exchange service. Therefore, all inputs of the transaction are saved. This is not done if one of the addresses is Shapeshifts main deposit address or an address of a trading platform. It is important to classify the new addresses into two different categories and so fill the two buckets described in step 1.

The first category includes single used addresses, which are used only for forwarding money once. They just have one incoming and one outgoing transaction. The second category contains multiple used addresses which are created by Shapeshift to merge UTXOs and have various incoming and outgoing transactions. So, if, e.g., after finding an outgoing transaction to an address identified as Shapeshift related, another outgoing transaction to the same address is found, it is classified as multiply used.

b. In case the output address is a P2SH address it is marked as a customer deposit. All outputs except the found one are removed, as they are not transactions to Shapeshift and won't be needed later.

4. Finally, we must delete the matching output address from the single address bucket in case it is contained in it. It won't be needed anymore as it is only one time used. In comparison, we never delete addresses from the other two buckets.

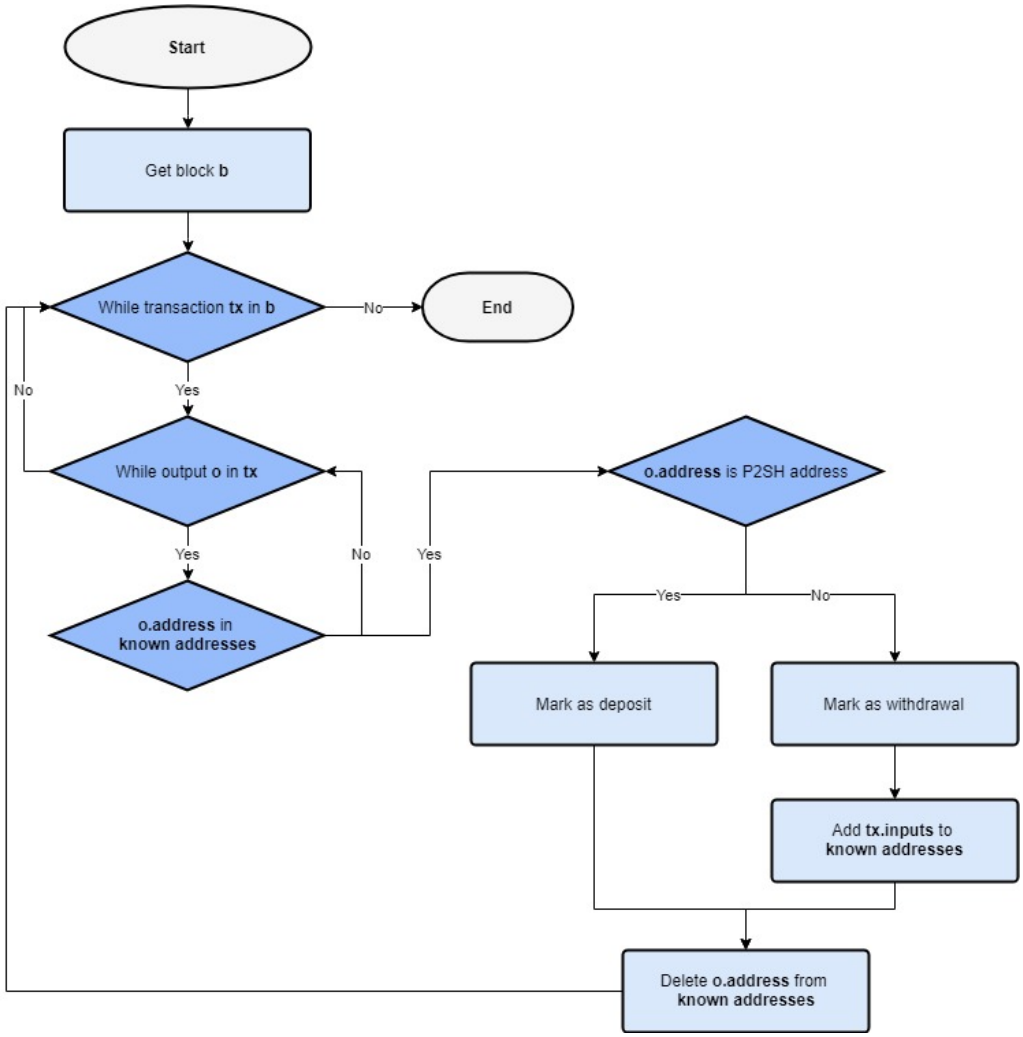


Figure 21: Address recognition Process for Bitcoin

This process is conducted for every transaction and guarantees that all exchange related transactions are found, in case they are connected to one of the external deposit addresses. To be sure that a particular trade is identified, the recognition process must start with the block where the transaction flow ends, more specifically where a

transaction to one of the external deposit addresses is made. As a transaction flow can expand over days, a high number of blocks should be proven beforehand. Figure 21 shows a simplified process flow of the recognition process for one block.

3.4.4 Recognition Process

Now we have defined multiple heuristics and will merge them into a single process. Checking every pair of transactions if they fit regarding the time difference and the exchange rate requires a high amount of comparisons and would be very inefficient. Therefore, it is more suitable first to reduce all transactions only to Shapeshift related ones. For this, we must retrieve blocks from the Ethereum and Bitcoin network and pass the transactions through the previously defined recognition algorithms. As seen, the algorithms, especially the one for Bitcoin, require passing through blocks in descending order. Therefore, the whole process must run the blockchains backward. At the beginning a fixed preparation range of blocks should be processed, only checking and saving Shapeshift related addresses and not considering the classification of transactions. This preparation is done until the blocks numbers with which the algorithm should start are reached. After this, the recognition algorithm can start. A simplified process flow is shown in Figure 22. The process can be subdivided as follows:

1. Blocks for a defined period are loaded from the Ethereum and Bitcoin blockchains. The number of blocks for Ethereum is much higher, as blocks are confirmed faster. For one day this would be 5,760 blocks (4 blocks/minute * 60 minutes * 24 hours) for Ethereum, assuming an average block confirmation time of 15 seconds, and 144 blocks (6 blocks/minute * 24 hours) for Bitcoin, taking an average confirmation time of 10 minutes. The process should start with blocks with the same confirmation time on both blockchains. Thus, it either can start with the current block numbers, but also can take earlier blocks. The second approach is recommended, as the recognition of addresses is only possible running the preparation process for multiple future blocks beforehand.

2. We now pass all transactions through the address recognition algorithm, gather all transactions which are Shapeshift related and classify them either as deposit or withdrawal. It has to be kept in mind that the found transactions are surely related to the exchange service, but it is not determined which cryptocurrency was involved in the deposit transaction.

3. For the comparison now, we need to separate the two given classes into two different arrays. Deposit transactions are loaded within their corresponding block to the first array. The blocks are then sorted by the block confirmation time, as we only need this time for the comparison. For withdrawals, we need the receiving time of the withdrawal transaction, which has the shortest time range to the confirmation time of the deposit. Therefore, the withdrawals are included in the second array separately and sorted by their transaction time.

4. Now, the deposit blocks are run in descending order and each is compared with the single withdrawal transactions. The time comparison can lead to three different behaviors:

    a. In case the timestamp for the block confirmation is higher than the timestamp for the receiving of the withdrawal, it is certain that this cannot be an exchange pair, as withdrawals can't happen before deposits weren't received. Since the blockchain is processed backwards the following withdrawals will have a smaller timestamp. Hence, the block with all contained transactions can be skipped, and the next block is checked.

    b. If the time difference is within a declared time range, said that all withdrawals are executed after the corresponding deposit is received and taking an upper bound into account, each transaction in the block is processed to be analyzed more in detail regarding the connection to the withdrawal.

    c. The last case left is when the difference between the confirmation time of a block and the time of a withdrawal surpasses the declared upper bound. The timestamps of the blocks will only get smaller due to the backward running process. Thus, the withdrawal transaction surely won't be matched with any other deposit and can be deleted. The time comparison of the block can then continue with the next withdrawal.

5. After a match is found in step 4b, all transactions in the deposit block can be compared with the withdrawal transaction regarding their values. First, the corresponding exchange rate between both currencies is calculated by retrieving their values in USD for the time the exchange was made. With this rate and an estimated Shapeshift fee, the expected height of the withdrawal for a given deposit amount is computed. This estimated value is then compared with each output of the withdrawal transaction. In case the value lies within the defined range, the two transactions are matched as a possible exchange pair and saved. The process allows either the deposit, as well as the withdrawal, to be included in multiple exchanges, as multiple Shapeshift related transactions can match to each other regarding time difference and exchange rate.

6. Finally, the block can be deleted, and the process can continue with the next one. After all blocks for the defined range of time has been analyzed, new previous blocks are loaded for the same range of time. This process is repeated until a set time limit is reached.

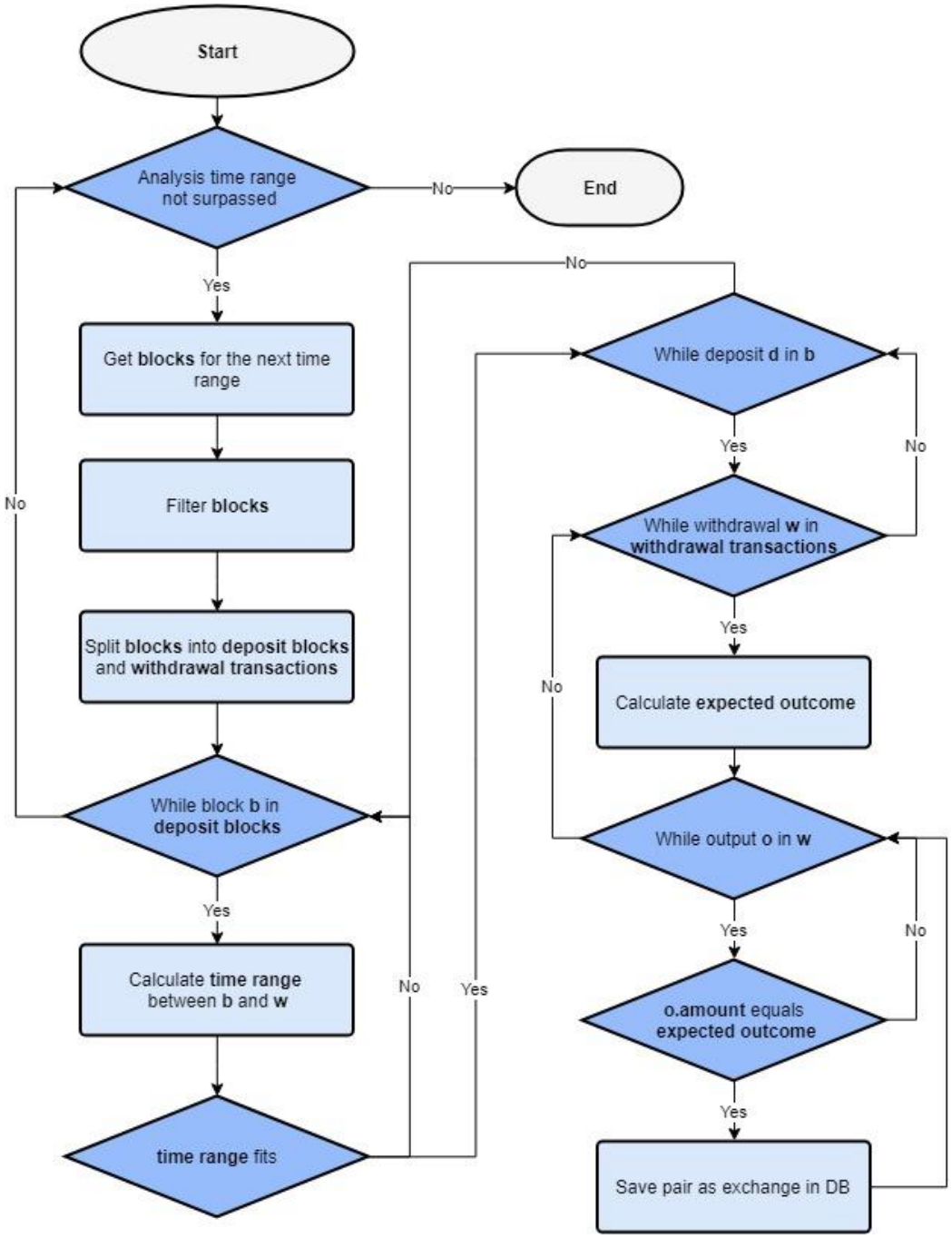


Figure 22: Cross-Blockchain Recognition Tool – Process Flow

3.5 Conception of the Evaluation

After running the previously defined algorithm, processes must be defined to evaluate how accurate the result is. We analyze the set of found exchanges in two different ways.

Comparison with Scraped Data

First, we compare the data for a given time range with the data, which was retrieved by the scraping algorithm for the same period. As a result, we can assign each exchange found by the recognition algorithm to one of the following four classes:

- True positive: A correct exchange was found by the algorithm, as well as by the scraper. As we can't assess if the scraped data contains all exchanges of a period, this category could also include real exchanges, which were found by the algorithm, but not by the scraper.
- False positive: A wrong exchange that was found by the algorithm, but not by the scraper.
- False negative: A correct exchange that was not found by the algorithm, but by the scraper. As we can't assess if the scraped data contains all exchanges of a period, this category could also include correct exchanges, which were neither found by the algorithm, nor by the scraper.
- True negative: This is a wrong exchange which was neither found by the algorithm, nor by the scraper and therefore is not visible in the output.

Our primary goal is to find all correct exchanges. This means having a high percentage of true positives and a low percentage of false negatives. Furthermore, it is also essential, to have high accuracy. Therefore, we want to have at best only the true corresponding withdrawal for one deposit and no other matching false withdrawals. This requires having a low percentage of false positives. After classification, we must further analyze why exchanges, found by the scraper, couldn't be found by the defined algorithm. Also, we must identify reasons, why transactions were wrongly matched.

Check with Shapeshift API

As explained before, it is not determined if the scraped data contains all exchanges done in the given time range. Because of this uncertainty, we should check each exchange that was found by the algorithm but not by the scraper, more in detail. Therefore, we send the deposit address of every exchange to the API of Shapeshift. The output returns which cryptocurrencies were involved in the exchange with this deposit address. If an exchange was made between Ethereum and Bitcoin, this represents an exchange, which was not found by the scraper. We then check if the corresponding withdrawal transaction was found by the tool. This could change the classification of exchanges after the first evaluation step. Some of the exchanges, which were classified as false positives, are therefore true positives, although they are not contained in the scraped data. Furthermore, after having more detailed information from the API response, like the currencies and the amounts involved, we can find out more in detail which reasons cause wrong matching.

3.6 Conception of the Data Provision

Requests

The last step will be to provide the found data through an API. We realize this by creating a REST API which handles different GET requests. Exchanges can be found using following parameters:

- input address of deposit transaction
- input address of withdrawal transaction
- deposit address
- withdrawal address
- hash of deposit transaction
- hash of withdrawal transaction
- block number and currency of the deposit transaction
- block number and currency of the withdrawal transaction
- range of time

We encounter a problem with the search for exchanges by input address. As we don't save input addresses of the transaction pairs, the transaction hashes in which this address was involved must be retrieved from an external service. Then it can be searched if any of the hashes is contained in the found exchanges. Depending on the sent input address, many transaction hashes might be found, e.g., if the input address is one of Shapeshifts four forwarding addresses for Ethereum. Therefore, the response must be limited. For the other GET requests involving addresses or hashes, the found data is checked regarding their existence. All matching exchanges are then returned. In the best case, there is only one pair in the response. In case there are multiple matching exchanges found for an address, the entries are ranked by the difference between the estimated outcome value and the output value of the withdrawal transaction. The last GET request returns all found exchange pairs in the requested range of time. The search is realized by looking at the block confirmation time of the deposit transaction, as this is the time that comes the closest to Shapeshifts timestamp.

Concept

To make such a REST API possible the requested data must already be saved in the database. Running the algorithm after every request would only allow searching for exchanges by providing a specific time of interest, but not by providing addresses or hashes. Furthermore, this would be slow and inefficient as the preparation requires passing through a high number of blocks first. Therefore, we load the exchanges before providing the REST API. To get historical exchange data the algorithm will run and analyze the given blockchains backward starting at a defined point in time. For future exchange

data, the scraper can be used, as well as the algorithm, which should analyze the earliest range of time periodically. The upside of using the scraper is that all found pairs are surely true. The downside is that we depend on the availability of the Shapeshift APIs. This is not the case for the recognition algorithm. Nevertheless, current exchanges can't be found immediately by the recognition algorithm due to the preparation range of the address recognition. A certain amount of time should pass by to be able to identify those exchanges. This can also require waiting a couple of days. Therefore, using the scraping algorithm for finding current exchanges seems more suitable.

All in all, the concept of the tool can be realized by running the scraper for getting current and future exchanges, while the recognizer algorithm can find all historical exchanges before the time the scraping started. The REST API can then be used to request both data sets at the same time.

4 Implementation

4.1 Environment

The implementation was realized with the programming language Python 2.7 and executed on an Ubuntu 17.10 server with four vCPUs and 16 GB RAM. All tool classes can be found in the main folder. The scraper specific classes are in the scraper folder and the implementation of the evaluation in the analysis folder. The file structure is shown in Figure 23. The contents of each file will be described in this chapter.

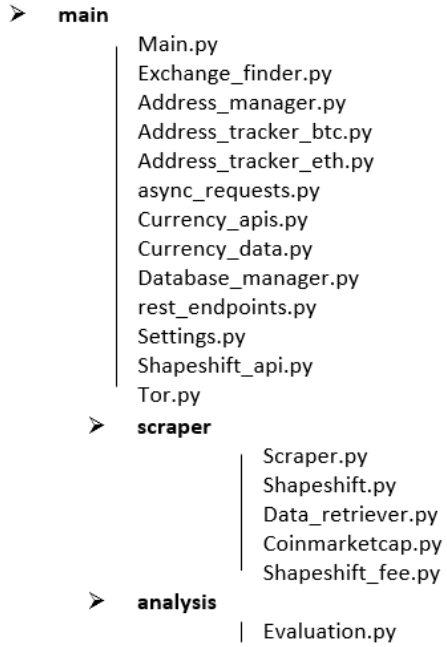


Figure 23: File Structure of the Project

Within the project following external services were used:

SERVICE NAME	USAGE
Shapeshift	Used for scraping data and evaluation
Infura	Procurement of Ethereum blockchain data
Blockchain.info	Procurement of Bitcoin blockchain data
Etherscan.Io	Procurement of the transaction history for the main Shapeshift address (Ethereum)
Cryptocompare	Procurement of historical exchange rates
Coinmarketcap	Procurement of actual exchange rates

4.2 Database Schema

Scraped Data

The data scraped by the scraping program is saved in a MySQL database with following columns:

COLUMN NAME	DESCRIPTION
id	Intern ID
currency_from	The input currency of the exchange
currency_to	The output currency of the exchange
amount_from	The amount of the input currency (deposit)
amount_to	The amount of the output currency (withdrawal)
fee_from	Transaction fee paid by the user for the input transaction
fee_to	Transaction fee paid by Shapeshift for the output transaction
fee_exchange	Exchange fee paid by the customer to Shapeshift
address_from	Shapeshift address to which the customer sends the deposit (deposit address)
address_to	Customer address to which Shapeshift sends the withdrawal amount (withdrawal address)
hash_from	Hash of the deposit transaction
hash_to	Hash of the withdrawal transaction
time_from	Time the deposit transaction was received in the network of the input currency
time_block_from	Time the deposit transaction was confirmed in a block
time__exchange	Time the transaction was tagged by Shapeshift as started
time_to	Time the withdrawal transaction was received at the network of the output currency
time_block_to	Time the withdrawal transaction was confirmed in a block
block_nr_from	Number of the block the deposit transaction was contained in
block_nr_to	Number of the block the withdrawal transaction was contained in
dollarvalue_from	Corresponding value of the input currency in US Dollar at the time of the exchange
dollarvalue_to	Corresponding value of the output currency in US Dollar at the time of the exchange

All exchanges are saved in the database regardless of their currency and are filled with the available data returned from the API and external services (*currency_from, currency_to, amount_from, fee_exchange, dollarvalue_from, dollarvalue_to*). Exchanges that have Bitcoin or Ether as input currency, furthermore contain the data retrieved from the blockchain and the Shapeshift API. These are all data points except *fee_to*, *time_to*, *time_block_to* and *block_nr_to*, which are only retrieved if also the output currency is either Bitcoin or Ether. For this work, we just focus on these two currencies. Anyways, with the scraped data, it is also possible to add additional data to the exchanges containing other cryptocurrencies by searching through the respective blockchain, just the same way it is done for the two currencies in this thesis.

Recognition Tool

The MySQL database schema for the data saved by the tool has almost the same structure as the scraper data. Only the Shapeshift exchange time (*time_exchange*) is not existent, as we lack this information. Furthermore, the column stating the exchanger fee does not contain the real Shapeshift fee, but the estimated one.

4.3 Implementation Details

4.3.1 External Services and Helper Classes

Multiple helper classes are responsible for a particular function within the search process. Some of them are used by both, the tool and the scraper.

Tor

We use APIs from different extern suppliers, and the problem arises that some of these services only allow a limited number of requests for a certain time range. To bypass this limitation, we use a Tor control library named stem to change our IP when required. For this, we must first enable the Control Port in the *torrc* (*etc/tor/torrc*) file and therefore set following parameters:

- ControlPort 9051: Port to be used by Tor
- CookieAuthentication 1: Authentication through generating a cookie. Users having permission to read the cookie file are allowed to use Tor. The file can be found at */var/run/tor/control.authcookie*.

Then the IP can be changed by calling the *change_ip* method (Figure 24) implemented in *Tor.py* which sends a signal to change the circuit.

```
1.  def change_ip():
2.      with Controller.from_port(port = 9051) as controller:
3.          controller.authenticate()
4.          controller.signal(Signal.NEWNYM)
```

Figure 24: Method for changing the IP Address

Transaction and Block Retriever

All methods used for retrieving data from the blockchains can be found in *Currency_apis.py* and are listed in the following table.

METHOD NAME	DESCRIPTION
get_last_block_number	With this method, the current block number can be requested for a given cryptocurrency. For Bitcoin, we use the Blockchain.info service, and for Ether, we request the Infura service. In case a request fails the IP is changed, and a new request is sent again a minute later.
get_block_by_number	Here, a block, with all the corresponding transactions, is loaded for a given currency and a given block number. For this purpose, we again use Blockchain.info and Infura. As blocks with Bitcoin transitions are returned without order, we sort them by the time they were received. In case of a request failure, the IP is changed and the request is repeated. Before a block is returned, it is standardized.
standardize	Used to put the information of a block into a standardized structure regardless of the currency. This standardization allows creating a recognition process that can be applied to every cryptocurrency. If new currencies should be integrated, we only must translate the given block data into this format.
get_transactions_for_address	Used to get transactions of a specified address needed for the tools API. For this, the Etherscan.io and Blockchain.info services are used.

The standardization results in a dictionary with following keys:

KEY	DESCRIPTION
symbol	Unique abbreviation of the cryptocurrency name, e.g. “BTC” for Bitcoin and “ETH” for Ether.
time	Time the transaction was received in the network.
blocktime	Time the transaction was confirmed in a block.
fee	Height of the fee paid for doing the transaction.
hash	Hash of the transaction.
block_nr	Number of the block the transaction was contained in.
inputs	Array containing all inputs of the transaction.
input.amount	Height of an input.
input.address	Address from which the transaction input originates.
outputs	Array containing all outputs of the transaction.
output.amount	Height of the output.
output.address	Address to which the transaction output was sent.
is_exchange_deposit	Boolean showing if the transaction is a deposit.
is_exchange_withdrawal	Boolean showing if the transaction is a withdrawal.

Currency Rate Retriever

To get exchange rates, we use two different extern services for the scraper and the tool.

For the scraping algorithm, we use the API of Coinmarketcap. The implementation can be found in the Coinmarketcap.py file. This service returns the current price in USD of all cryptocurrencies listed on the website. To avoid a high request load, the response with all prices is saved and can be then retrieved by the scraper by requesting the price for the needed currency. A new request to Coinmarketcap is sent every 10 minutes to keep the rates up to date.

For the recognition algorithm, we need historical data. For this, we use the service of *Cryptocompare*. It allows to retrieve price information for the last seven days minutely and all prices before that hourly. Therefore, in the implementation, the hourly API is used. Just as with the previous service, the retrieved data is saved and provided to the tool when needed. As we run the blockchains backward, the price data is also requested for descending timestamps. The API allows getting the last 2,000 data points before a given time. Thus, when the tool starts searching for historical exchanges the time for the first transaction is passed to this API, retrieving the price data for the time of the given transaction, as well as the prices for the last 2,000 hours. The algorithm uses this data for the following transactions which lie within this range. When this limit is surpassed, new data is retrieved again. Every time the saved price data for a given time is requested by the tool, different checks are made. First, all saved price data that lies more than one hour in the future from the time the tool is currently analyzing is deleted, as it won't be needed anymore. If the deleting process leads to an empty data set, new data is loaded. After that, the data point for the requested time is retrieved, and the price is calculated, by getting the mean between the highest recorded and the lowest recorded value in the given hour.

Shapeshift Data Retriever

The file *Shapeshift_api.py* contains all requests used to get data from the public Shapeshift API. Following methods are used:

METHOD NAME	DESCRIPTION
get_exchange	Method to get detail information about an exchange passing the deposit address as input.
get_fees_shapeshift	Returns the Shapeshift exchange fees for all cryptocurrencies traded on the platform.
get_exchanges_shapeshift	Returns the last 50 exchanges executed by the service.

As the scraper uses this APIs often, the IP address is changed before each of these requests.

Database Manager

The Database Manager, which can be found in the *Database_manager.py* file, handles all communication with the MySQL database. The Manager includes following methods:

METHOD NAME	DESCRIPTION
initialize_db	Used at the start time of the tool. First, the database which will contain the found exchanges is created. This is done over the *create_database*-method. After that, a static class is created which connects to this database and is responsible for handling requests to it. Therefore, it provides methods for executing queries and committing changes to the database. As problems were encountered due to losing connection to the database after a certain amount of time, all calls run through a process which reestablishes the connection in case it was lost.
create_database	Connects to the MySQL server and creates the database.
create_table_exchanges	Creates the database table for the data found by the tool according to the model presented in 4.2.
create_table_scraper	Creates the database table for the scraped data according to the model presented in 4.2.
insert_exchange	Inserts and commits the data of one found exchange into the database.
insert_multiple_exchanges	Inserts and commits the data of multiple found exchanges into the database within one command.
insert_shapeshift_exchange	Inserts the scraped data retrieved from the Shapeshift API. Here, only the initial data is set before searching through the blockchains.
get_shapeshift_exchanges_by_currency	Returns all exchanges for a given currency for which no additional data from a blockchain was found yet.
update_shapeshift_exchange	Updates a certain exchange entry for which additional data was found on the blockchain.
update_shapeshift_exchange_corresponding_tx	Updates a certain exchange entry for which additional data for the outgoing transaction was found.
delete_all_data	Deletes all data found by the recognition algorithm.
delete_all_scraper_data	Deletes all data found by the scraper.

Settings

There are different values for the scraper, as well as for the recognition algorithm, which can be adapted before running and influence the precision of the programs. These parameters can be found in Settings.py. As the parameters depend on which cryptocurrency is analyzed they are handled over methods which return different values regarding of the currency inputted. The methods include:

METHOD NAME	DESCRIPTION
get_rate_lower_bound	The percentage a withdrawal transaction amount at least must have from the expected amount to be matched to a deposit transaction. The default value is 90%.
get_rate_upper_bound	The maximal percentage a withdrawal transaction amount can have from the expected amount to be matched to a deposit transaction. The default value is 110%
get_exchange_time _lower_bound	The minimum value the time interval between the block confirmation time of the deposit and the transaction time of the withdrawal is allowed to have to put the pair into further analysis. The default value is 0 (minutes).
get_exchange_time _upper_bound	The maximum value the time interval between the block confirmation time of the deposit and the transaction time of the withdrawal is allowed to have to put the pair into further analysis. The default value is 15 (minutes).
get_preparation_range	The number of blocks to be checked before starting the search process in order to build up a dataset of address needed for the address recognition.
get_exchanger_fee	The estimated Shapeshift fee for a given transaction
get_scraper_offset	The number of blocks to be skipped when beginning a new search loop in the scraping algorithm.
get_scraper_offset _last_block	The number of blocks to be analyzed after a given limit in the search loop of the scraping algorithm. The limit is always the starting block number of the previous search loop.
get_scraper_offset_ for_first_iteration	The number of blocks to be analyzed in the first search loop of the scraping algorithm, as no limit was set yet.
get_block_number _for_hour	The average number of blocks confirmed within an hour for a certain currency. We assume an average confirmation time of 10 minutes for Bitcoin and 15 seconds for Ethereum. Therefore, six blocks are set for Bitcoin and 240 for Ethereum.

4.3.2 Implementation of the Shapeshift Scraper

The Scraper structure is shown in Figure 25. It can be divided into three main parts.

Scraper Main

The starting point for running the scraper is the *Scraper.py* file. From here the main method is called, which first creates the database, established the connection and creates the table in which the exchanges will be saved. After that, two processes are run in parallel. The main process is the retrieving of the 50 last transactions from Shapeshift. For this, the *Shapeshift* class is created and executed every 30 seconds. For the second process,

a new thread is created, in which the Finder class is involved, that handles the finding of the additional information.

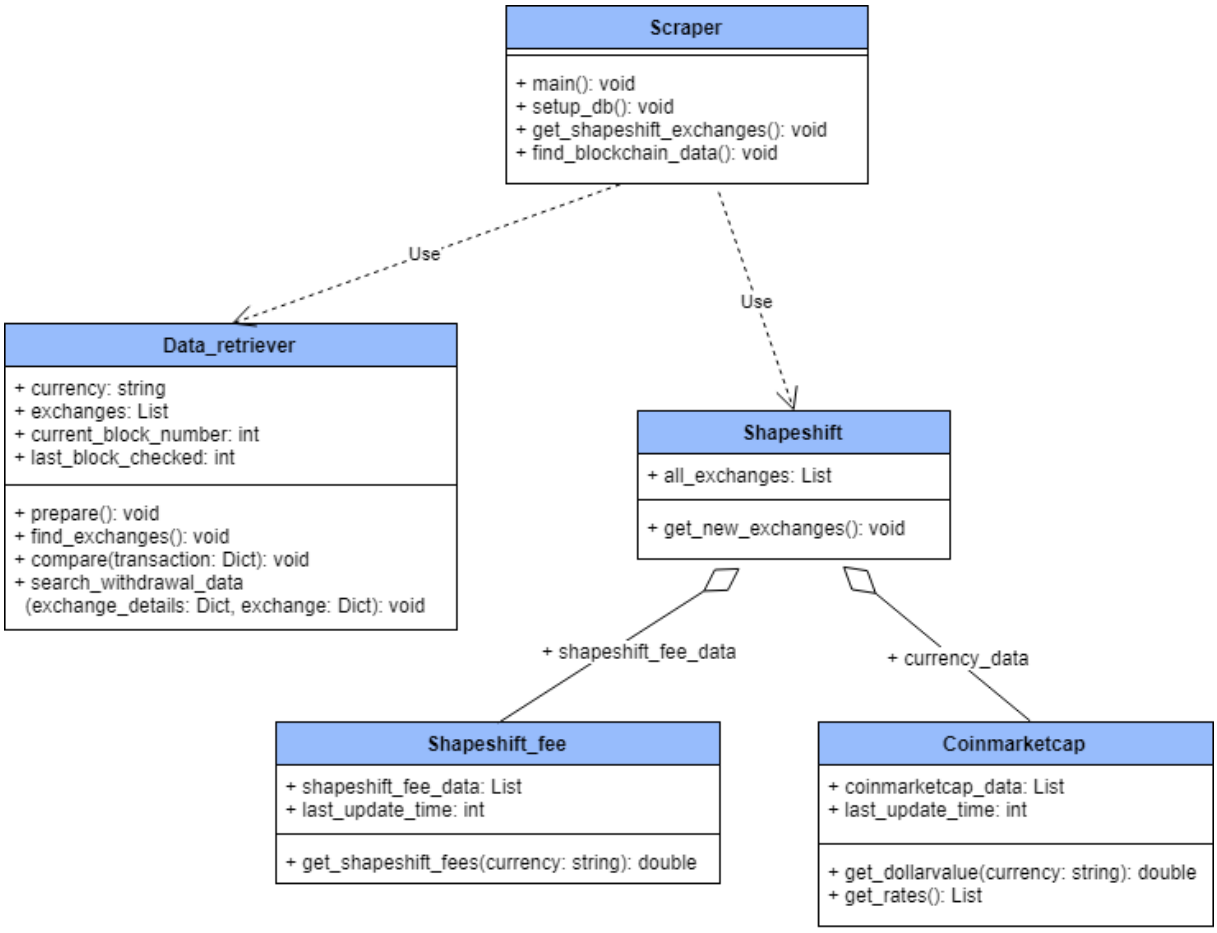


Figure 25: Scraper - Class Diagram

Retrieving New Shapeshift Exchanges

The *Shapeshift* class in *Shapeshift.py* holds an array with all exchanges retrieved from the Shapeshift API, which were not saved to the database yet. Every time the main method *get_new_exchanges* is called, two processes are executed sequentially.

First, the newest 50 exchanges are retrieved sending a request to the Shapeshift API. It is checked if the exchanges were already retrieved before, by comparing them to the previously mentioned array. New exchanges are added to this array. Finally, all exchanges are sorted by their timestamp.

Then, it is checked if there are exchanges in the array which are over one minute older than the oldest exchange of the last 50 retrieved exchanges. These won't be needed for a later comparison anymore. Therefore, they are saved to the database, together with the value of the involved currencies in US Dollar and the Shapeshift fee, and deleted from the array. For the database, the order of the exchanges is reversed so that they are sorted in ascending order of their timestamp. The values in USD are retrieved by using the Coinmarketcap class, previously described. The Shapeshift fees are retrieved from the *Shapeshift_fee* class (located in *Shapeshift.py*), which requests and saves the fees for all currencies every 30 minutes.

Finding Additional Data

The find method creates a *Data_retriever* class for every currency and runs them every 30 minutes. Each of these classes is responsible for searching for additional data for exchanges with incoming transactions of the given currency.

First, all exchanges, which were not found yet, are retrieved from the database, as well as the current block number for the given currency. This is done in the prepare method.

After this preparation, the searching process starts by calling the method *find_exchanges*. For this, blocks are retrieved for a given range. This range starts few blocks before the current block number. This offset is used, as the exchanges to be analyzed are not current, but happened a few minutes ago and so some blocks can be skipped. The offset height is defined in the settings for every cryptocurrency. The end of the range is the starting block number of the previous analysis loop minus another offset also determined for every currency in the settings. Here the offset is needed, as new exchanges might have been added lately by Shapeshift and be included in blocks, which were already checked.

In the compare method, we iterate over all contained transactions of a block and compare each with every exchange from the list. Both lists are passed through in reversed order, meaning going back in time. Two timeframes are calculated for each pair:

- The interval between the confirmation of the block and the Shapeshift timestamp
- The timespan between the time the transaction was received in the network and the Shapeshift timestamp

With these calculated intervals, different checks are run. The second defined timespan is expected to be very small, as Shapeshift sets its timestamp shortly after receiving the transaction in the network. This means the time span should be positive and low. Therefore, we should stop checking a blockchain transaction when this interval gets negative. As Infura doesn't return the time an Ethereum transaction was received, the range is allowed to be negative going down to 10 minutes of difference, to prevent the case an exchange would not be recognized if the confirmation took long. If the timespan is higher than this lower bound and smaller than 6 minutes, the pair is taken to further analysis. The last case uses the interval involving the block confirmation time. If this is higher than 10 minutes the currently analyzed exchange can be removed from the list, as the range is too high to declare this pair as an exchange and the blocks to follow will make this interval even higher.

After this comparison, the possible exchange pairs are analyzed further. If the amount of any output matches the amount of the exchange, we check if passing the address to the Shapeshift API returns additional information. If the response has the exchange status "complete" and the currencies and the amount matches the ones of the exchange, the exchange can be considered as found. Therefore, the exchange is updated with the additional data, the corresponding transaction is searched, and the exchange is removed from the list. The search for additional data of the outgoing transaction is done in the *search_withdrawal_data* method. Just as in the main process, the data is retrieved from

Infura or Blockchain.info. The exchange is then updated again with this additional information.

4.3.3 Implementation of the Recognition Tool

Main

A class diagram for the tool implementation is shown in Figure 27. The recognition process starts in Main.py. In the main method, the database with all tables is set up, and all cryptocurrencies which should be involved are declared together with their starting block numbers. Then the parameters are passed to the *Exchange_finder* class, and the search is started.

Finder

The main process of the *Exchange_finder* class is executed in the *find_exchanges* method (Figure 26). After initializing all needed parameters and helper classes, the class starts the preparation algorithm of the Address Manager and then loads the first block for every currency to be analyzed. This is done to get the historical time from which the search will be executed, as at the beginning only block numbers are available. The smallest timestamp is defined as the start point.

```
def find_exchanges(self):
    #Preparation
    self.address_manager.prepare(self.current_block_number_dict)
    self.load_first_blocks()
    range_to_analyze = self.hours_whole_analysis * (60 * 60)
    analysis_time_range = self.hours_single_loop * (60 * 60)
    start_time = self.get_min_blocktime()
    current_search_time = start_time

    #Search
    while start_time - current_search_time < range_to_analyze:
        current_search_time = current_search_time - analysis_time_range
        self.load_blocks(current_search_time)
        for block_from in list(self.blocks_from):
            current_block_time = block_from[0]["blocktime"]
            newest_transaction_time = self.transactions_to[0]["time"]
            if current_block_time < datetime.datetime.utcfromtimestamp(current_search_time):
                break
            elif current_block_time > newest_transaction_time:
                self.blocks_from.remove(block_from)
            else:
                self.delete_old_withdrawals(block_from)
                self.async_comparing(block_from)
                self.blocks_from.remove(block_from)
        self.save_found_exchanges()
```

Figure 26: Recognition Tool - Main method

After this, a time limit is set until which the analysis will proceed. Until the reaching of this point loops, with a fixed time range to be analyzed, are sequentially run. In each loop, blocks are loaded for both currencies until a block is retrieved, which surpassed the given time range. To speed up the process, the requests are sent asynchronously, which is realized in the *load_blocks* method. Here, for every currency, an estimated number of blocks for the set time range is requested. It is then checked again if more block must the loaded, in case the range wasn't surpassed yet. Within this process, all transactions are

passed to the *Address_manager* class, which filters them and only returns Shapeshift related transactions. The filtering is handled in different threads for each currency in the

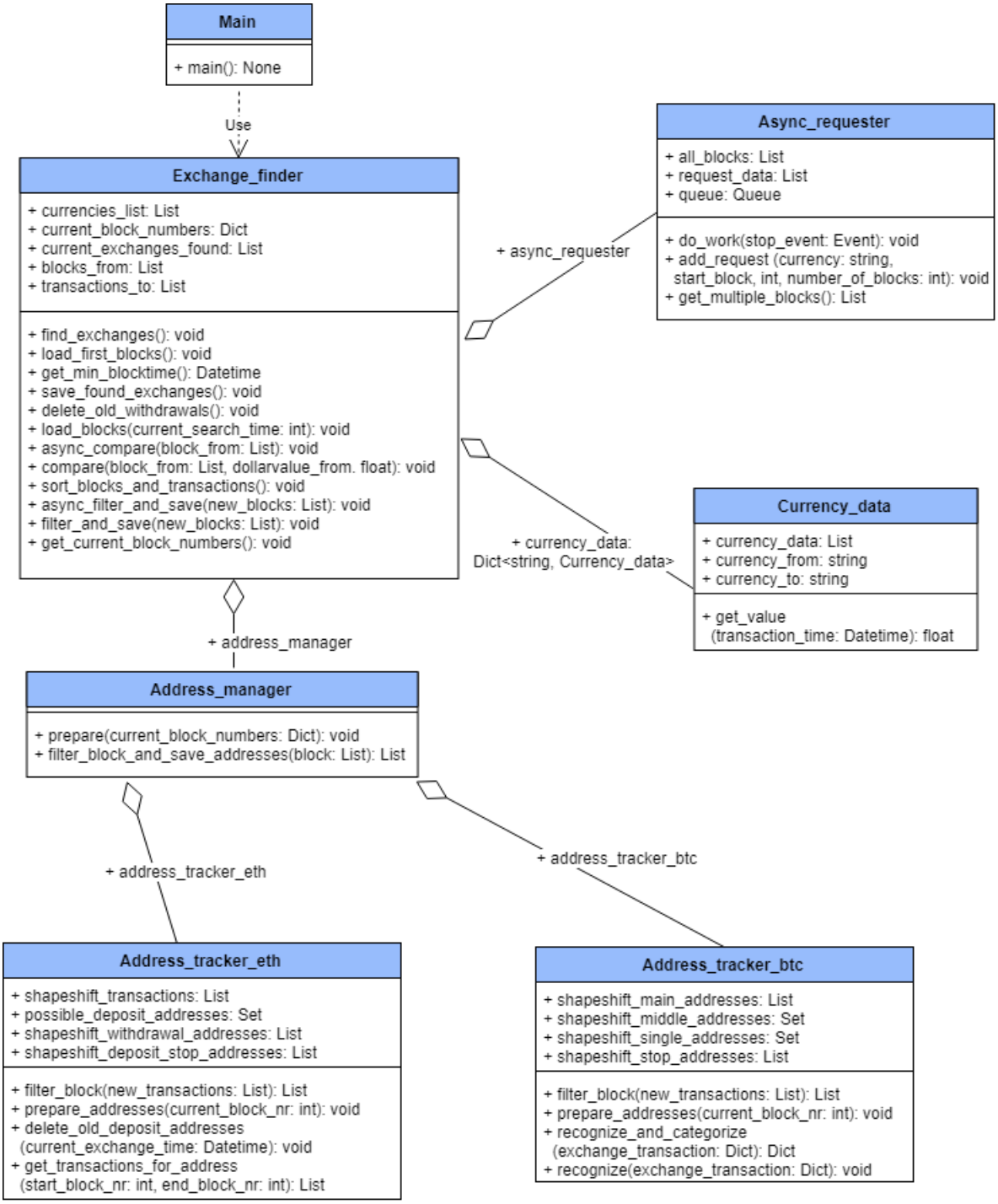


Figure 27: Recognition Tool - Class Diagram

filter_all method. Finally, the marked transactions are split up into two separate arrays depending on their classification. Deposits are appended within their block to the *blocks_from* array and withdrawals separately to the *transactions_to* array.

Now that we have all data from the blockchains, the comparison starts. First, we iterate over the sorted blocks in *blocks_from*, containing the deposit transactions. For each, it is checked if its timestamp surpassed the time range. In this case, the comparison would

stop, and new blocks would be retrieved for the next time range. Furthermore, we check if the block confirmation time is higher than the time of the newest withdrawal. As it is not possible that the deposit was made after this withdrawal the block can be deleted, and we can proceed to the next one. If a block passed these two checks, the transactions it holds are compared with the withdrawal transactions. First, all withdrawals, which happened a certain amount of time after the confirmation time of the block, are deleted, as they won't be needed anymore (*delete_old_withdrawals*). Then, each deposit is compared with the possible withdrawal transactions. This is done asynchronously in the *async_comparing* method. After the comparison, the block is deleted, and the next one is checked. After all blocks of the current analyzed time range were screened, the found possible exchanges are saved to the database at the same time (*save_found_exchanges*), and the process can continue with the following loop.

The previously mentioned comparison of one deposit with all withdrawals is made in the compare method. First, the time difference for the possible exchange is calculated sequentially for every withdrawal. As long as this timespan doesn't fall below the lower bound declared in the settings, the exchange rate is calculated for the two involved currencies using the US Dollar value retrieved from the *Currency_data* class. After also getting the Shapeshift fee from the settings the expected outcome resulting from the deposit amount can be stated. The last step then is to iterate over all outputs of the withdrawal transaction and check if the amount is within the declared range around the expected outcome. In case of a match, the data for the found pair is stored in the *current_exchanges_found* list.

Address Manager

The *Address_manager* class creates the specific address recognition classes for each currency and offers the two methods prepare, used at the very beginning of the recognition algorithm in order to build up the address data set, and *filter_block_and_save_addresses*, which categorizes transactions as Shapeshift related and builds up the address data set with help of the given block. Both methods pass the values to the corresponding class depending on the currency of the given block.

The filtering of Ethereum blocks is realized in the *Address_tracker_eth* class. Here the *prepare_addresses* method implements the retrieval of all Shapeshift transactions done 1.5 days after the starting point by using the Etherscan.io API. This way we don't have to retrieve blocks which wouldn't be analyzed any further and speed up the process. First, the initial Ethereum block is requested to get its block confirmation time. With this, we then retrieve all transactions where Shapeshift was involved as a receiver and do this until the defined time limit is reached. After the preparation is done and the address data set was built up, the *filter_block* method can be used for every newly downloaded block. Here, first, all Shapeshift addresses, which surpassed the time limit of 1,5 days and won't be used anymore, are deleted from the address list. Then each transaction from the received block goes through some checks. First, the output address is compared with the Shapeshift main address. If these values match, the transaction is added to the address data set. Otherwise, it is checked if the transaction is a deposit or a withdrawal. A deposit

is recognized by finding its output address in the list of saved addresses (*possible_deposit_addresses*) and a withdrawal by checking if the input address is one of the four known sending addresses of Shapeshift (*shapeshift_withdrawal_addresses*). The marked Shapeshift transactions are then returned as a block.

The *Address_tracker_btc* class handles the filtering of all Bitcoin blocks. It also has two main methods. Firstly, the *prepare_addresses* method, which builds up the Shapeshift address data set for Bitcoin. Here, we retrieve all blocks, starting with the block which is the defined number of blocks higher than the initial block number. For every block, we check if the transactions can be connected to Shapeshift. We do this until the initial block is reached again. The recognition follows the same pattern as described in chapter 3.4.3. For this, multiple lists are used. The *shapeshift_main_addresses* list contains all external deposit addresses, *shapeshift_middle_addresses* lists all addresses which are used multiple times by the exchange service and *shapeshift_single_addresses* saves all Shapeshift addresses used only once. The *shapeshift_stop_addresses* list furthermore contains Shapeshifts main deposit address. The second main method is the *filter_block* method, which iterates over all transaction of a received block and checks if the transaction is related to the exchange service by going through the same procedure as in the recognition process of the preparation. The only difference is that the found transactions are marked either as deposit or withdrawal, as this will be needed later for the comparison.

Asynchronous Requests

As mentioned before we retrieve multiple blocks at the same time to speed up the process. For this, the *Async_requester* class is used. It holds the *request_data_list*, which contains the information of which blocks have to be retrieved. It is filled through calling the *add_request_data* method, which adds a dictionary to the list, containing the currency, a block number, and the number of blocks that should be download starting from the given block number. The requests are sent when the *get_multiple_blocks* method is called. Here a defined number of threads are started at the same time, which then retrieve values from a queue. In the next step, this queue is filled with all data added to list, and all requests are sent asynchronously by the different threads. Each retrieved block is saved in the *all_blocks* list. Finally, we wait until all threads are done, sort the final list of blocks and return it.

4.3.4 Implementation of the Evaluation Process

The Evaluation of the implemented algorithm happens in the Evaluation.py file. Before starting, the output data must be downloaded from the database in Excel file format. The same must be done for the scraped data. Now, having these two files, the *run_whole_analysis* method is triggered, to start the evaluation. First, the entries from the two files are loaded into data frames, and the entries of the dataset of the recognition algorithm (*df_found_data*) are filtered to have the same time range as the scraped data

(*df_scraped_data*). With this, the two evaluation methods, described in the conception are executed.

First, the two data frames are compared with each other to find entries with the same values. For this, we examine if the addresses and hashes match. If they do, this is marked in both data frames by setting a Boolean value to true. After analyzing all entries, the marked data is written into new Excel files, and the marked data frame containing the tool data is returned for further analysis.

With the returned data frame, the *find_with_shapeshift_api* method is called, which evaluates if the found exchanges are right although the scraper didn't recognize them. For this, two new columns are created in the data frame. One column documents the real output currency of the exchange, which is returned from the Shapeshift API. With this, we can later check which currency was the corresponding output for a deposit of a wrongly marked exchange. Also, if no output currency is returned, it can be assumed that the trade with the given deposit address was unsuccessful and no withdrawal transaction was executed. The second column is a Boolean value which marks an exchange pair that was correctly found by the tool but not by the scraper. The evaluation process now iterates over all entries, which are grouped by their deposit addresses. Throughout the process, all entries with the same deposit address are saved into a list. This procedure is done until an entry with a different deposit address is encountered. In case there is no entry which was already found by the scraper the group is further checked. For this, the deposit address is sent to the Shapeshift API. We expect to get back a result as we know the deposit address is a real Shapeshift address. As there may be unsuccessful exchanges, we check if the result contains an output currency. If given, the output currency is added to all entries of the group. If the currency is Ether or Bitcoin, we know that the deposit address is involved in an exchange between these two currencies. Therefore, we can check if any of the entries contains the data of the correct withdrawal transactions, by comparing the address and the hash. A match is marked, and the analysis continues the evaluation for all the following groups. At the end, the data frame is exported as Excel file again.

4.3.5 Implementation of the REST API

The implementation of the REST API can be found in the rest_endpoints.py file. The interface is realized using the Flask web framework. We defined multiple routes, which offer the request possibilities described in the conception. The routes are listed in the table below.

SEARCH BY	ROUTE
Deposit address	/address_from/$address
Withdrawal address	/address_to/$address
Deposit transaction hash	/hash_from/$hash
Withdrawal transaction hash	/hash_to/$hash
Customer sending address	/input_from/$address
Shapeshift sending address	/input_to/$address

Block numbers & incoming currency	/block_nr_from? currency=$currency &start=$blocknumber &end=$blocknumber
Block numbers & outgoing currency	/block_nr_to? currency=$currency &start=$blocknumber &end=$blocknumber
Block confirmation time	/time_range? start=$datetime &end=$datetime

All queries are executed through the method *query_db*, which gets the desired result from the database and defines the structure of the returned JSON string. This structure contains all parameters from the database and the additional field *diff_from_expected_outcome*, which contains the amount the found exchange differentiates from the expected value. With this, the result list can be sorted. The smaller the amount is, the more probable it is that the exchange is correct.

5 Evaluation

The evaluation of the accuracy of the tool was conducted for exchange data within a time range of one week (23.02.2018 00:00 - 01.03.2018 23:59). First, general information about the scraped and the tool data are presented. Afterward, the comparison of both sets is fulfilled.

Scraped Data

For the given time range we could scrape data with following general information:

PARAMETER	VALUE
TOTAL EXCHANGES	64,688 Exchanges
VALUE RECEIVED	48,039,179.40 $
TOTAL FEES RECEIVED	70,346,89 $
AVERAGE FEE PAID PER EXCHANGE	1.09 $

Figure 28 shows the hourly number of exchanges over this week.

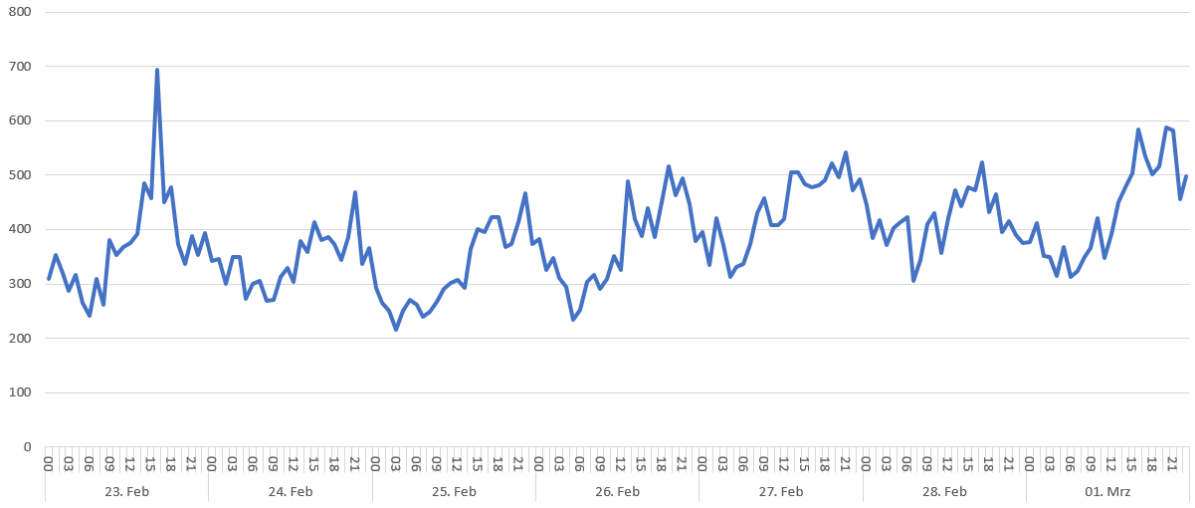


Figure 28: Scraped Data - Time Distribution

We take a closer look at which currencies were involved in the exchanges. Figure 29 shows the most popular currencies to send and the most popular to receive. The scraped data contains 49 different deposit and 50 different withdrawal currencies.

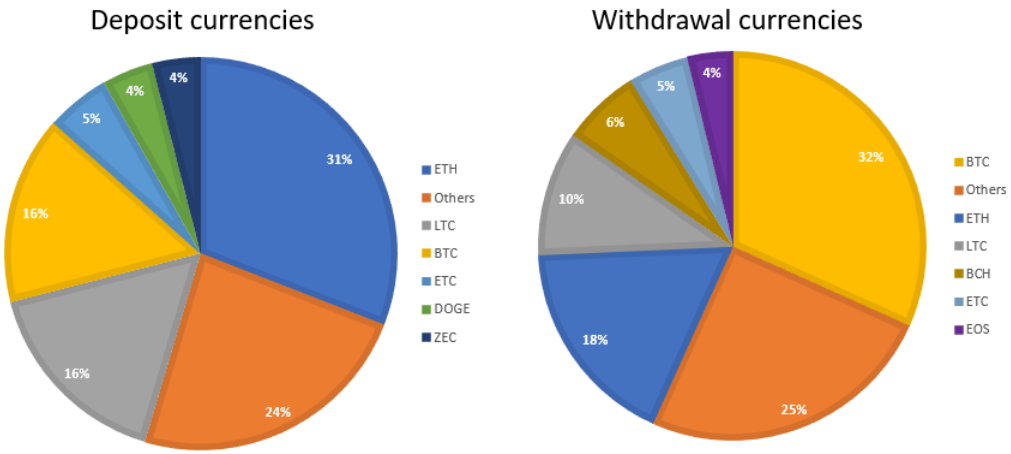


Figure 29: Scraped Data - Currency Occurrence

Figure 30 presents the most popular exchange pairs in the analyzed time range. In total, 1,228 different pairs were counted. The two most frequently executed exchanges were between Ether and Bitcoin.

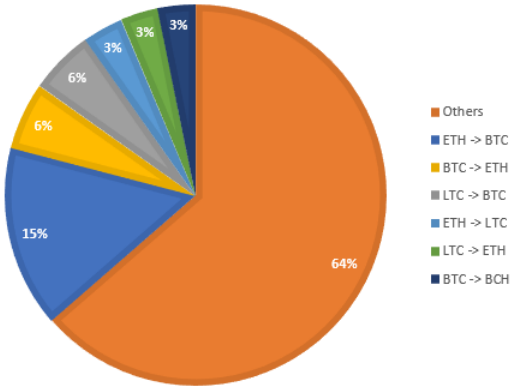


Figure 30: Scraped Data – Currency Pairs Occurrence

The dataset contains 9,972 exchanges from Ether to Bitcoin and 3,689 from Bitcoin to Ether. This corresponds 21% of all scraped exchanges. For all these exchanges, the scraper searched for additional data on the blockchains. For 647 Exchanges (around 4.7%) no data could be found. A reason could be, e.g., that an address was multiple times used by the customer and thus it is not possible anymore to check if data found on the blockchain is the right one. This means we have 13,014 exchanges executed between Ether and Bitcoin in the defined time range, which we try to find by using the recognition algorithm.

Tool Data

We ran the recognition tool assuming a Shapeshift processing time of maximally 15 minutes and allowing a deviation of the withdrawal amount from an expected amount by 10%. This had an output of 76,271 possible exchanges. In total, 28,138 different deposit addresses were matched. This means each deposit was matched to 2.7 withdrawals in average. The output data furthermore contains 20,569 different withdrawal addresses. As shown in Figure 31, more exchanges from Ether to Bitcoin were found than vice versa.

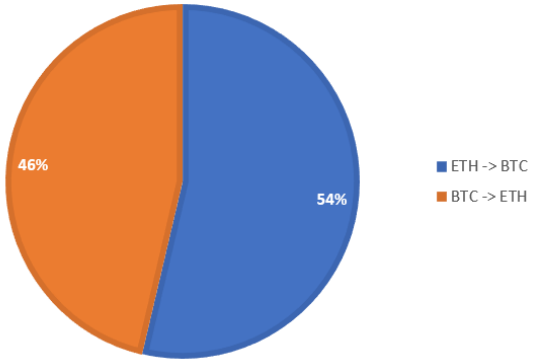


Figure 31: Tool Data - Currency Pairs Occurrence

Comparison

The comparison of both data sets showed that the tool could find 11,936 exchanges (true positives) out of the 13,014 exchanges documented by the scraper. This means 1,078 exchanges were not found (false negatives). Furthermore, 64,335 additional transaction pairs were matched, which are not contained in the scraper dataset (false positives). Therefore, the tools rate for correct matches is 91,7%, while its accuracy lies by around 15,6%. These rates are also visualized in Figure 32.

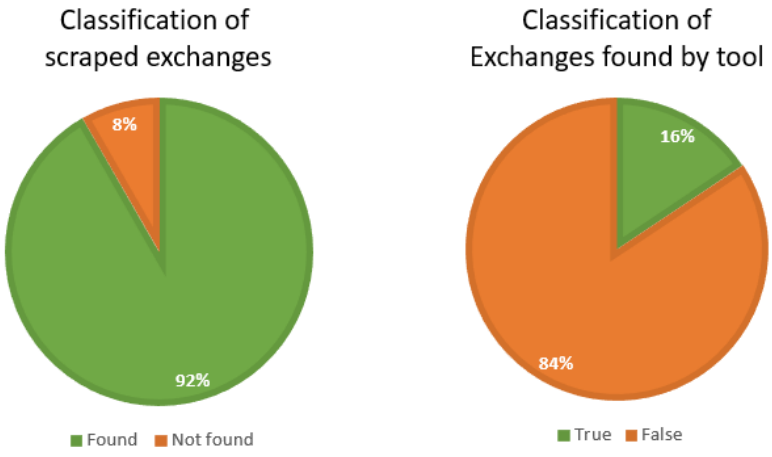


Figure 32: Evaluation - Result

We could identify multiple reasons that explain why some exchanges couldn't be found:

First, this can be due to the small amount involved in the exchange. As explained before, the probability of calculating a proper expected amount for a small deposit gets lower the

smaller a deposit is. Bitcoin deposits smaller than 0.0005 BTC were mainly not found, as the Shapeshift fee is almost as high as the withdrawal amount. Thus, not having the exact exchange rate and the real Shapeshift fee height, the expected amount can lie far away from the actual withdrawal amount. Furthermore, we can observe that small Ether deposits, which lie under 0.01, are mostly not forwarded to Shapeshifts main address, and therefore weren't found.

This observation is part of the next identified problem. Deposits are sometimes not forwarded, or this is done very late. Thus, the address recognition won't determine a deposit address as Shapeshift related. For the Bitcoin address recognition this problem is also encountered for withdrawals. They can be only recognized if they are contained in a flow of transactions ending in one of the external deposit addresses. A detailed analysis of the tool data showed that this problem affects most of the exchanges which couldn't be found. There are two reasons the involved Bitcoin addresses were not recognized as Shapeshift related.

On the one hand, this happens because the flow didn't end in an external deposit address at the time the analysis had started. The here evaluated data resulted from the execution of the tool with a preparation range of 5,000 Block for the Bitcoin address recognition. Setting a lower preparation range results in a decreased amount of found real exchanges, as many flows end after a high number of blocks. This means that the larger the preparation range is, the more exchanges can be found. So, some of the exchanges might not have been found as the transaction flow they are contained in didn't end in the defined 5,000 blocks.

On the other hand, the flows ended in addresses, which weren't identified before, such as new generated storage or unknown external deposit addresses. Transactions contained in a transaction flow ending in such a temporary address can't be found, as long as this address is not known. In the manual analysis of Shapeshifts transaction flows on the Bitcoin network we identified two addresses, which couldn't be classified to a specific owner. Not knowing these addresses would reduce the accuracy of the tool. We can assume that other similar addresses exist that couldn't be identified, and this lack of knowledge leads to the inability of finding exchange related addresses.

Furthermore, we want to show where the high amount of false exchange pairs comes from. This result can be explained by the fact that Shapeshift executes many transactions with similar values in a short time range on each blockchain. Each of these transactions is connected to a deposit or withdrawal transaction on a different blockchain. As we can't determine which currencies were involved, this results in a high number of matches.

The result could be improved by setting the range for the expected value lower. Nevertheless, this would decline the number of real exchanges found. The table below shows the results for different ranges and proves that a smaller range leads to a higher percentage of true exchanges, but also reduces the amount of these.

range	TOTAL	FOUND	TRUE
-10% to 10%	76,271	91.7%	15.6%
-5% to 5%	52,693	90.4%	22.3%
-2% to 2%	35,530	79.0%	28.9%
-1% to 1%	22,114	56.4%	33.2%

Another factor influencing the number of correctly matched exchanges is the defined range for the processing time. The default range reaches from 0 to 15 minutes. Setting a higher scale would lead to more correct matches, as the scraped data shows that there are exchanges over the default limit. Nevertheless, it would also increase the number of false positives. The table below shows the results for different ranges.

LIMIT	TOTAL	FOUND	TRUE
15 minutes	76,271	91.72%	15.6%
10 minutes	60,289	91,68%	19.8%
5 minutes	41,645	76.71%	24.0%
2 minutes	23,631	25.95%	14.3%

Check by API

In the second step, all deposit addresses were sent to the Shapeshift API to assess the outcome more in detail.

The evaluation showed that the tool found 2,616 correct matches, which were not recognized by the scraper. This finding improves the percentage of real exchanges found to 19.1%. The reason why the scraper did not detect these exchanges is the high confirmation time of the deposit. The additionally found trades have an average deposit confirmation time of almost 28 minutes. Apparently, Shapeshift adds exchanges to the list of last executed exchanges after a deposit was confirmed. The timestamp Shapeshift assigns to an exchange corresponds the time shortly after receiving the deposit transaction in the network. The additionally found exchanges have such a high range between these two times that at the time they could have been added by Shapeshift to the last executed exchanges, the intern Shapeshift timestamp already had surpassed other 50 exchanges with a more recent timestamp. Therefore, this data was never provided through the public API and couldn't be scrapped. This problem concerns mostly Bitcoin deposits, as this currency has a high average confirmation time.

The evaluation through the Shapeshift API furthermore allowed us to check the real withdrawal currency for every deposit. The result showed that the deposit address of 60.7% of all found exchanges was really involved in exchanges between Ether and Bitcoin. 37.0% of the exchanges had a different withdrawal currency involved. The remaining 2.3% were involved in unsuccessful exchanges. These are, e.g., deposits which surpass the allowed exchange amount range. Mostly they were marked as "resolved". These are

deposits which could be refunded to the customer. Several were also marked as “failed”, as no refund could be sent due to a missing refund address.

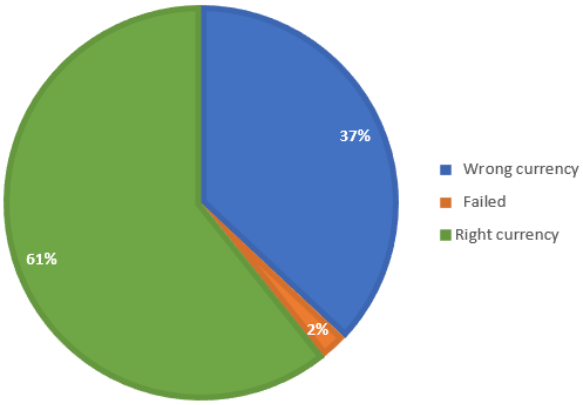


Figure 33: Evaluation of Currency Assignment

Conclusion of the Evaluation

The designed algorithm found most of the exchanges that were documented by the scraper. Regarding the not found exchanges, the biggest issue was encountered by the recognition of Shapeshift related addresses on the Bitcoin network. The complex and unstructured transaction structure makes it difficult to uncover all relevant addresses. For Ether transactions, Shapeshift uses a more simple and transparent money proceeding structure. Therefore, almost all addresses can be found here. If two Shapeshift transactions belonging to one exchange were identified, they very likely get matched, as most of the transaction pairs fulfill the previously described ranges for time and rate. The dimensions of the ranges influence how many correct exchanges will be identified, as well as the number of matches in total, including wrong ones. Therefore, appropriate ranges must be set.

Additionally, we detected exchanges, which were not provided by Shapeshift through their API. These were mostly exchanges containing a Bitcoin deposit which took a long time to be confirmed. Due to the in average large block confirmation time of 10 minutes on the Bitcoin network, this currency is mainly affected.

Lastly, we want to discuss the accuracy of the tool. All transactions involved in the algorithm are surely deposit or withdrawal transactions of a Shapeshift exchange. Nevertheless, a high number of wrong exchanges are composed. The reason for this is that many transactions are part of exchanges involving other cryptocurrencies. As we are not able to detect the real currencies involved, these transactions are also matched. Furthermore, many deposits and withdrawals with similar amounts of money are transferred at almost the same time. Without having the real exchange rate and Shapeshift fee, a deposit can’t be assigned to a withdrawal for sure.

As the tool matches multiple deposits to multiple withdrawals, no assurance can be given about the validity of a found exchange. Nevertheless, the tool aims to help to recognize the right transaction pair by ranking the exchanges.

6 Conclusion

6.1 Findings

Finally, we want to outline all findings of this thesis and shortly sum up the outcome of the three research questions presented at the beginning.

RQ 1: What is the Current State of the Art regarding Cryptocurrency Exchange?

First, we categorized cryptocurrency exchanges into trading platforms and over-the-counter markets and instant cryptocurrency exchanges. The characteristics of these services were explained in detail and compared with each other. Then the general process of instant cryptocurrency exchanges was analyzed using Shapeshift and Changelly as an example. Additionally, we took a look at current projects aiming to improve the exchange process by removing the intermediary.

RQ2: How can Cross-Blockchain Transactions be recognized?

After giving an overview of the available blockchain data analysis tools and explaining the terminology needed for understanding the exchange processes, we started with the conception of the recognition tool. Firstly, we showed how data can be retrieved by exploiting Shapeshifts API. Based on these data we identified different parameters with which we established heuristics. For this purpose, also the transaction flows of Shapeshift on the Bitcoin and Ethereum network were analyzed in detail, determining addresses related to the service. All heuristics were brought together to implement a tool, which can identify cross-blockchain transactions.

RQ3: How accurate is the implemented Solution? What are the Limits?

After the implementation of the designed algorithm, we evaluated the result of the tool. We compared real exchange data retrieved by the scraper with the exchanges matched by the recognition algorithm. The output showed that most of the scraped exchanges could be found by the tool. We stated the possible reasons why we couldn't identify all trades and explained why additional exchanges, which were not found by the scraper, were discovered. Finally, we described why transactions were matched wrongly by the tool. All in all, the evaluation depicted that the implemented recognition algorithm is able to find cross-blockchain transactions, but also includes the matching of wrong transaction pairs.

6.2 Outlook

We showed in this work which processes must be taken to establish heuristics and implement an analysis tool based on these. Further implementations are possible to improve the tools outcome. Such an improvement can, e.g., be the integration of data from more blockchains allowing the recognition of exchanges between more cryptocurrencies. For this purpose, the block data of each blockchain must be transformed to the defined structure in the tool, and an address recognition for the given blockchain must be implemented.

Many intents have been done to combine multiple addresses on one cryptocurrency network to one user. The data retrieved in this work gives a base to fulfill such matching of addresses over various blockchains. Nevertheless, we cannot determine that the sender of an exchange is the same person as the receiver, because the customer can also send the exchanged amount to the wallet of another person. Therefore, more investigation must be taken to guarantee a reliable algorithm.

As blockchain-based technologies will continue to expand into different areas, more stakeholders will have interest in analyzing the data stored on the ledger. This will also concern data related to cross-blockchain transactions, especially as there are many different efforts to improve this kind of trades currently. The process in this thesis showed how the recognition of such transactions can be realized and can be used to implement comparable algorithms for the identification of cross-blockchain of other services.

7 Bibliography

Alqassem, I., & Svetinovic, D. (2014). Towards reference architecture for cryptocurrencies: Bitcoin architectural analysis. In Internet of Things (iThings), 2014 IEEE International Conference on, and Green Computing and Communications (GreenCom), IEEE and Cyber, Physical and Social Computing (CPSCom), IEEE (pp. 436-443). IEEE.

Bitcoin average confirmation time (2018). Retrieved 15 04, 2018, from: https://blockchain.info/de/charts/median-confirmation-time

Bitcoin confirmation (2018). Retrieved 15 04, 2018, from: https://en.bitcoin.it/wiki/Confirmation

Bitcoin-otc (2018). Main Page. Retrieved 15 04, 2018, from: https://bitcoin-otc.com/

BitcoinVisualizer (2018). BitcoinVisualizer Github Project. Retrieved 15 04, 2018, from: https://github.com/thallium205/BitcoinVisualizer

Bitcoinwiki (2018). Comparison of exchanges. Retrieved 15 04, 2018, from: https://en.bitcoin.it/wiki/Comparison_of_exchanges

Bitfinex API (2018). Retrieved 15 04, 2018, from: https://docs.bitfinex.com/v1/reference

Bitfinex Cold Wallet (2018). Retrieved 15 04, 2018, from: https://support.bitfinex.com/hc/en-us/articles/213892469-How-secure-is-Bitfinex-

BitInfoCharts (2018). Retrieved 15 04, 2018, from: https://bitinfocharts.com/bitcoin/

BitIodine (2018). BitIodine Github Project. Retrieved 15 04, 2018, from: https://github.com/mikispag/bitiodine

Bitquick (2018). How-To. Retrieved 15 04, 2018, from: https://www.bitquick.co/how-to

Bittrex API (2018). Retrieved 15 04, 2018, from: https://bittrex.com/home/api

Blockchain.io (2018). Main Website. Retrieved 15 04, 2018, from: https://blockchain.info/

Blockchain.io Statistics (2018). Blockchain Size. Retrieved 15 04, 2018, from: https://blockchain.info/de/charts/blocks-size

Blocktrades.us (2018). Main Website. Retrieved 15 04, 2018, from: https://blocktrades.us/

BTCSpark (2018). BTCSpark Github Project. Retrieved 15 04, 2018, from: https://github.com/JeremyRubin/BTCSpark

Buterin, V. (2014). Ethereum: A next-generation smart contract and decentralized application platform.

Chainalysis (2018). Main Website. Retrieved 15 04, 2018, from: https://www.chainalysis.com/

Changelly (2018). Main Website. Retrieved 15 04, 2018, from: https://changelly.com/about

Changelly API (2018). Retrieved 15 04, 2018, from: https://changelly.com/developers

Coinmarketcap (2018). Cryptocurrency Market Capitalizations. Retrieved 15 04, 2018, from https://coinmarketcap.com

Coinswitch (2018). Main Website. Retrieved 15 04, 2018, from: https://coinswitch.co/

Coinswitch (2018). Track Order. Retrieved 15 04, 2018, from: https://www.coinswitch.co/app/track

Duong, T., Chepurnoy, A., & Zhou, H. S. (2018). Multi-mode Cryptocurrency Systems.

Elliptic (2018). Main Website. Retrieved 15 04, 2018, from: https://www.elliptic.co/

Ethereum Avarage Gas Limit (2018). Retrieved 15 04, 2018, from: https://etherscan.io/chart/gaslimit

Ethereum Average Block Time (2018). Retrieved 15 04, 2018, from: https://etherscan.io/chart/blocktime

Ethereum Github (2018). Ethereum Clients. Retrieved 15 04, 2018, from: https://github.com/ethereum/wiki/wiki/Clients

Etherscan.io (2018). Ethereum ChainData size. Retrieved 15 04, 2018, from: https://etherscan.io/chart2/chaindatasizefast

Evercoin (2018). FAQ. Retrieved 15 04, 2018, from: https://evercoin.com/faq

Evercoin (2018). Main Website. Retrieved 15 04, 2018, from: https://evercoin.com/

Exchanges Volumes. Retrieved 15 04, 2018, from: https://coinmarketcap.com/exchanges/volume/24-hour/

Frunza, M. (2015). Solving Modern Crime in Financial Markets: Analytics and Case Studies.

Galitskiy, V., Shpin, P., & Virk, R. (2015). Online Automatic Auctions for Bitcoin Over-The-Counter Trading, White paper.

Herlihy, M. (2018). Atomic Cross-Chain Swaps.

Hevner, A., March, S.T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. In: MIS Quarterly, (28)1, 75-105.

Hosp, J., Hoenisch, T., & Kittiwongsunthorn, P. (2018). COMIT. Cryptographically-secure Off-chain Multi-asset Instant Transaction network. Making global payments as cheap, fast and easy as sending a text message, v1.0.2

Infura.io (2018). Documentation. Retrieved 15 04, 2018, from: https://infura.io/docs

ItBit (2018). OTC exchanges. Retrieved 15 04, 2018, from: https://www.itbit.com/otc#starttrading

Kalodner, H., Goldfeder, S., Chator, A., Möser, M., & Narayanan, A. (2017). BlockSci: Design and applications of a blockchain analysis platform.

Lansky, J. (2016). Analysis of Cryptocurrencies Price Development. In: Acta Informatica Pragensia, 5(2), 118-137.

Leung, A. (2016). Shapeshift Rebuilds After Losing $230,000, Promised to Be Back Wednesday. Retrieved 15 04, 2018, from: https://cointelegraph.com/news/shapeshift-rebuilds-after-losing-230000-promised-to-be-back-wednesday

LocalBitcoin (2018). How-To-Use. Retrieved 15 04, 2018, from: https://localbitcoins.com/guides/how-to-sell-bitcoins-online

Moeser, M. (2013). Anonymity of Bitcoin Transactions. An Analysis of Mixing Services

Moore, T., & Christin, M. (2013). Beware the Middleman: Empirical Analysis of Bitcoin-Exchange Risk

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). Bitcoin and Cryptocurrency Technologies. Princeton University Press

Ploom, T. (2016). Blockchains–wichtige Fragen aus IT-Sicht. Blockchain Technology (S. 123-147). Berlin, Deutschland: De Gruyter Oldenburg.

Poon, J., & Dryja, T. (2016). The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments

Richest Bitcoin Addresses (2018). Retrieved 15 04, 2018, from: https://bitinfocharts.com/top-100-richest-bitcoin-addresses.html

Shapeshift (2018). Main Website. Retrieved 15 04, 2018, from: https://info.shapeshift.io

Shapeshift API (2018). Retrieved 15 04, 2018, from: https://info.shapeshift.io/api

Tschorsch, F., & Scheuermann, B. (2016). Bitcoin and beyond: A technical survey on decentralized digital currencies. IEEE Communications Surveys & Tutorials, 18(3), 2084-2123.

Vaishnavi, V., Kuechler, W., and Petter, S. (2017). Design Science research in Information Systems.

Wood, G. (2014). Ethereum: A secure decentralised generalised transaction ledger. Ethereum Project Yellow Paper, 151, 1-32.