Technische Universität Berlin

Detecting Liquidity Draining "Rug-Pull" Patterns in CPMM Cryptocurrency Exchanges

by

Grigori Akimov

Matriculation Number 408290

A thesis submitted to

Technische Universität Berlin
School IV - Electrical Engineering and Computer Science
Department of Telecommunication Systems
Service-centric Networking

Master's Thesis

December 13, 2021

Supervised by:
Prof. Dr. Axel Küpper

Assistant supervisor:
Prof. Dr.-Ing. habil. Magedanz

SNET
SERVICE-CENTRIC NETWORKING

Eidestattliche Erklärung / Statutory Declaration

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed.

Berlin, December 13, 2021 Grigori Akimov

Abstract

As the cryptoasset market continues its unparalleled growth, new solutions that facilitate the exchange of cryptoassets emerge. Many decentralized exchanges innovate to be fully automated and permissionless entities that enable the exchange of cryptoassets without the need for an order book. However, these emerging technologies are prone to exploits by malicious actors. An ever increasing number of investors become victims of scams known as *rug-pulls*. In light of these developments, we dedicate this work to the examination of rug-pulls on the Ethereum ledger. First, we provide a formal definition for the term rug-pull. Subsequently, we develop an on-chain heuristic for the identification of rug-pulls. Using our heuristic we analyse the on-chain data from *Uniswap V2* which is one of the most popular decentralized cryptoasset exchanges in order to measure the extent of rug-pulls there. Finally, we also propose a series of rug-pull preventing measures.

Zusammenfassung

Während der Markt für Kryptowährungen stetig weiter wächst, entstehen neue Lösungen für den Handel mit Kryptowährungen. Viele der dezentralisierten Kryptobörsen sind vollautomatisierte Entitäten, die den Handel mit Kryptowährung ohne ein Auftragsbuch möglich machen. Diese neuartigen Technologien werden jedoch auch für betrügerische Aktivitäten genutzt. Immer mehr Marktteilnehmern werden Opfer einer betrügerischen Aktivität, die allgemein unter dem Namen *rug-pull* bekannt ist. Vor dem Hintergrund dieser Entwicklungen, ist nachfolgende Arbeit der Untersuchung von rug-pulls auf Ethereum gewidmet. Zuerst wird der Begriff rug-pull formal definiert. Anschließend, erfolgt die Entwicklung einer on-chain Heuristik, die rug-pulls identifizieren kann. Mit dieser Heuristik werden Daten von der dezentralisierten Kryptobörse *Uniswap V2* untersucht und das Ausmaß des rug-pull Phänomens gemessen. Zum Schluß wird eine Liste an Maßnahmen vorgeschlagen, die der Vorbeugung von rug-pulls dienen können.

Abbreviations

AMM - Automated Market Maker
CEX - Centralized Exchange
CPI - Consumer Price Index
CMMM - Constant Mean Market Maker
CPMM - Constant Product Market Maker
CSMM - Constant Sum Market Maker
CPU - Computational Processing Unit
DAPP - Decentralized Application
DEFI - Decentralized Finance
DEX - decentralized exchange
DOS - Denial Of Service
EOA - Externally Owned Account
ETH - Ether
ERC20 - Ethereum Request for Comment 20
EVM - Ethereum Virtual Machine
FATF - Financial Action Task Force
FED - United States Federal Reserve
GPU - Graphical Processing Unit
KYC - Know Your Customer
LP - Liquidity Provider
P2P - Peer to Peer
PoS - Proof of Stake
PoW - Proof of Work
QE - Quantitative Easing
USD - United States Dollar
WETH - Wrapped Ether

Contents

1 Introduction

"What is needed is an electronic payment system based on cryptographic proof instead of trust."

—Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2008

For thousands of years financial systems across the world have remained overwhelmingly *custodial*. Even before the invention of money itself, ancient Babylon already relies on centralized institutions in the form of temples to keep track of credits issued and debts owed[1, p.15]. In the subsequent millennia further foundational concepts are introduced such as the invention of paper money in China or the establishment of central banking in England before the modern financial system begins to take shape at the end of the 19th century. However, the risks associated with relying on centralized institutions materialize themselves long before the age of modern finance. For instance, only a few centuries after the Chinese Empire innovates from gold and silver coins to paper money, its currency faces catastrophic hyperinflation as the ruling dynasty decides to issue ever more bills in order to finance its war efforts[2]. Other notable examples of extreme inflation include the devaluation of the Roman *Denarius* during the 3rd century crisis or the *Kipper and Wipper* financial crisis at the beginning of the 30-years-war[3]. Looking at recent history, the years 1933 and 1971 are of high significance as the United States government first abolishes the right of creditors to exchange their Dollar bills for gold and 38 year later applies the same action to foreign governments effectively abandoning the *gold standard* of the US Dollar[4][5]. Insofar the US Dollar of today is a *fiat* currency not backed by any physical countervalue which in turn allows the United States Federal Reserve to increase the circulating supply of USD at will. This policy of increasing the monetary supply in times of crisis is known as *Quantitative Easing* and is being applied to soothe the economic effects of the current COVID pandemic[6][7]. Simultaneously, the purchasing power of the Dollar measured through the consumer price index continues to decline so that 100 USD from January 2008 are worth 125,49 USD in March 2021[8].

The seemingly untouchable narrative of centralized custodians being the only way to establish trust in a financial system is challenged by a roundmail distributed to a few hundred receivers in the spring of 2009. Herein the pseudonym *Satoshi Nakamoto* describes a P2P non-inflatable money system with the concept of an electronic coin as a chain of cryptographic signatures in its center[9]. Decentralized in its nature, *Bitcoin* possesses all basic components of a blockchain as illustrated in figure 1.1 and thus does not rely on a custodian to verify the legitimacy of transactions. Instead following the principle of *asymmetric cryptography* each transaction is signed with the private key of the sender and the public key of the receiver so that the integrity of the chain is verifiable by anyone at any time[9]. To govern *transactions* and to facilitate a *consensus mechanism* in a non-custodial system, Nakamoto introduces *Proof-of-Work*. Through the PoW algorithm nodes of the Bitcoin network are incentivized to verify all broadcasted candidate transactions in their vicinity and compress them into blocks of legitimate transactions

which after further verification are added to the *distributed ledger*[10, p.180-200]. Insofar, Bitcoin represents the first of its kind in a novel, previously impossible class of financial assets. The interconnected and continuously growing global computational infrastructure allows for the emergence of non-custodial, decentralized blockchains known as *cryptocurrencies* and while Bitcoin is the first, many others follow.

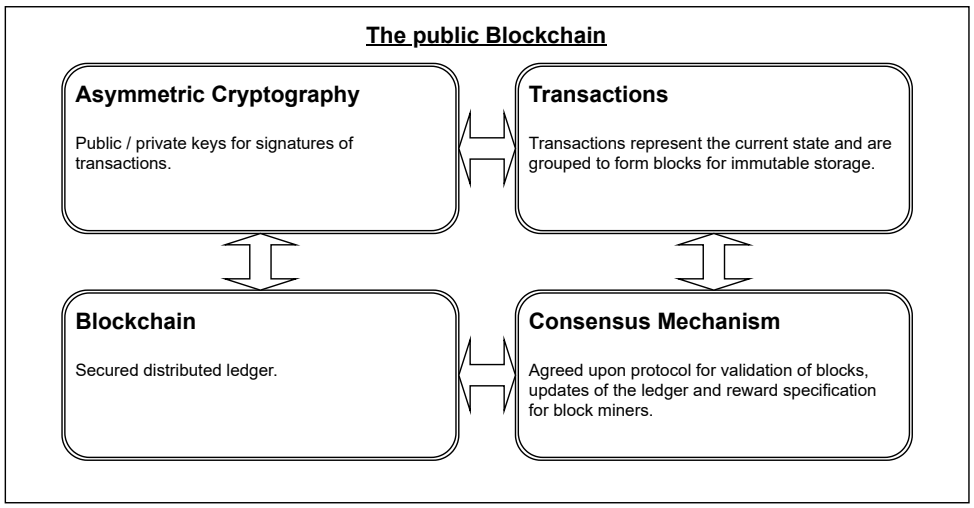

Figure 1.1: The basic components of a public blockchain[11].

7 years after the introduction of Bitcoin, the genesis block of a new public blockchain known as *Ethereum* is mined. However, unlike Bitcoin, Ethereum is not only storing transactions in its blocks, but also the *state* of the network[12]. Changes to the state are performed through the execution of so-called *smart contracts* via the turing-complete *Ethereum Virtual Machine*. These algorithmic constructs enable the deployment of distinct cryptoassets or *tokens* to run on the Ethereum blockchain. As a consequence of this feature a number of Ethereum-based, decentralized platforms emerge that facilitate the exchange of cryptoassets and that are known as decentralized exchanges or *DEXs*[13][14]. A sub-set of such platforms are so-called Automated Market Makers *AMMs*. These are implementations of decentralized exchanges that circumvent the need for an order book or counter-parties by relying on a deterministic pricing formula governed by a smart contract[14][15]. One of the possible models to implement an AMM is the Constant Product Model[14] where the resulting exchange is called a Constant Product Market Maker *CPMM*. A CPMM trading pair stores pooled reserves of two assets, and provides liquidity for those two assets maintaining the invariant following which the product of the reserves cannot decrease[16]. Since no order books exist, market participants are incentivised to place their assets within a CPMM liquidity pool by being entitled to a pro-rata share of the transaction fees[15]. Typically, liquidity providers are required to place an equal amount of both assets into the pools in order to maintain the exchange rate[17][15]. In conclusion a CPMM trading pair involves three roles: a trader who seeks to exchange assets, a liquidity provider who seeks to incur transaction fees and a governing smart contract that holds the assets in pools, sets the exchange rate and executes the exchange.
Unfortunately, emerging technologies such as CPMMs, also offer novel opportunities for malicious behaviour. Let us assume the following, simplified scenario: A liquidity provider who

provides the overwhelming majority of the liquidity decides to remove all of the provided liquidity at once and then immediately exchanges the returned cryptoasset for Ether, the native cryptocurrency of the Ethereum blockchain, commonly referred to as*ETH*. The quantity of tokens in the pools will decrease significantly and the consequent token exchange will result in a neigh total depletion of the ETH pool. Depending on Ethereum's network congestion, the necessary transactions can be completed within seconds leaving other holders with an effectively worthless cryptoasset.
The set of actions described above is publicly known as a *rug-pull*. Some rug-pulls are of malicious nature, where a liquidity provider simply disregards all other trading-pair participants[18]. Other rug-pulls are of fraudulent nature, where an ERC20-cryptocurrency is created with the sole purpose of being subjected to a rug-pull sometime in the future[19]. There are reports that alone in the months of September and October 2020, millions of US Dollars worth of Ethereum have been lost to rug-pulls[18][19]. In light of the apparent severity of the issue, this Master Thesis seeks to illuminate such malicious and fraudulent patterns in the context of ERC20-based CPMM cryptocurrency exchanges by utilizing on-chain analysis of the Ethereum public ledger.

2 Background

Before diving into the details of concept and design, this chapter illuminates the important background information. First, a high level overview over the Ethereum blockchain will be provided, followed by a detailed specification of smart contracts and CPMMs.

2.1 Ethereum

In 2013 *Vitalik Buterin* publishes the *Ethereum White Paper* a document proposing a distributed ledger capable of being much more than just an electronic payment system. The idea for Ethereum originates from Buterins efforts to improve *Mastercoin*, an overlay protocol for Bitcoin upon which distinct cryptoassets with distinct rules can be build[20]. Since these efforts remain fruitless, Buterin develops the first draft of Ethereum with the understanding that the layers running on top of the base protocol could be generalized[21]. Later this vision evolves from Ethereum being programmable money to a generic, decentralized computing platform[21]. So in its essence Ethereum is supposed to be a *world computer* in the sense that it "is a deterministic but practically unbounded state-machine with two basic functions; the first being a globally accessible singleton state, and the second being a virtual machine that applies changes to that state"[22, p.16]. Herein, the *Ethereum Virtual Machine* (EVM) is turing-complete meaning that it is theoretically able to solve any computational problem e.g. execute any program[23]. Another consequence of the turing-completeness lies in the fact that a state in Ethereum can be anything that can be represented on a modern computer[12]. The significant differences between a general purpose computer and Ethereum lie firstly in the fact that the state is saved on a *distributed ledger* instead of centralized memory and secondly in the fact that state changes are governed by a *consensus mechanism*[22, p.21]. Besides that, *transactions* on Ethereum are stored in blocks. However, unlike Bitcoin blocks, an Ethereum block does not only contain a list of transactions but also information about the most recent state[23]. Finally, *asymmetric cryptography* is used for the signature of transactions which do not necessarily need to include a transfer of tokens, but can also just signal a change of state. So in conclusion Ethereum contains all basic components of a blockchain as seen in figure 1.1.
Projects of such scale and complexity generally do not allow for a completion within a single setting, but rather require a years-long development cycle. In the case of Ethereum, the road map is divided into four stages with each stage containing multiple sub-releases known as *hard-forks* that introduce changes which are not necessarily backward compatible[22, p.19]. The initial stage called *Frontier* constitutes to the launch of the network itself in the year 2015 including the start of mining operations, listings on cryptocurrency exchanges, establishing a testing environment and allowing for the upload and execution of programs on the Ethereum network[24, p.177]. One year later the second stage known as *Homestead* is implemented. The major improvements here are the removal of network halts, an increase of the transaction speed

and several other preparations for future upgrades[25]. In October 2017 Ethereum implements its third release named *Metropolis* including an abstraction of security features, a simplification of the protocol and client implementations and further improvement of transaction speed[26]. The final and most extensive release known as *Serenity* introduces fundamental changes to the Ethereum ledger. Serenity's release begins in December 2020 and is planned to be completed in the course of approximately two years. The most profound change is a switch of the consensus algorithm from *Ehash*, an adaptation of Bitcoins *PoW*, to Ethereums own Proof-of-Stake *PoS* approach[25]. In order to achieve that transition multiple sub-releases such as *sharding* or *beacon chains* are implemented. However, at the time of the writing of this work, only the *beacon chains* have been released so far.

2.2 Components of Ethereum

Following section 2.1, it is clear that the complexity of Ethereum is so high that it is not feasible to describe its every part in detail. Therefore, we will only provide a high-level overview over components that are not directly relevant to the subject of this work. Nevertheless, we will inspect relevant components, specifically smart contracts, in detail. Finally, the context of our work is the current state of Ethereum's development and implications of planned future developments will be only considered marginally.

2.2.1 Consensus Mechanism

Ethereum currently implements an adaption of the Proof-Of-Work algorithm. Essentially miners compete with one another using computational power to be the first one to solve a hash function in order to produce a new block and collect the reward. Unlike Bitcoin, the Ethereum mining algorithm is better suited for GPUs instead of CPUs. In an effort to avoid mining centralisation, miners are able to reap partial rewards for mining orphaned blocks which is enabled due to shorter block times with the rewards being issued at a constant rate[27]. The last point implies that the Ethereum's ledger cryptocurrency *Ether* does not have a capped or maximum supply which is a fundamental difference to Bitcoin. As for Ethereum's block architecture, it is divided into three parts. First comes the block header containing crucial information about the block and its relation to the rest of the network, followed by a series of transactions and finally a list of blocks originating from the parent's parent block[12]. This list enables miners to reap rewards for mining orphaned blocks known as *uncle blocks* or *ommers*[27].

2.2.2 Accounts

In order to enable operations on the Ethereum ledger the concept of accounts is introduced with two distinct account types. The first is a so-called *external* or *user* account known as an *EOA*. They have an Ether balance, are secured by public–private key pairs and can send transactions, but these transactions are initiated by an external user or users and not by the account itself[28, p.28]. On the other hand, there are *contract* accounts that are governed by code instead of being controlled by a human. They can send transactions or change their state depending on information from new inbound transactions[27]. An Ethereum account object contains the following information[28, p.30]:

- **Nonce value**: Number of outgoing transactions or contract creations performed by the account.
- **Balance**: Account balance annotated in Ether.
- **Codehash**: Bytecode version of executable code in contract accounts; remains empty for external accounts.
- **Storageroot**: Root of the Merkle Patricia tree that contains code and data stored on the blockchain.

The value *storageRoot* is needed so that the 160-bit account addresses can be mapped to their respective states as the primary data-structure for key-value mapping in Ethereum is the Merkle Patricia tree[27].

2.2.3 Ethereum Virtual Machine and Smart Contracts

As described above *contract* accounts contain executable bytecode that is able to change the state of the network. To facilitate the code execution, a formal execution model is introduced that specifies the altered state of the system given inbound bytecode instructions[22, p.9]. This execution model is called the Ethereum Virtual Machine or *EVM* and in effect it is the emulation of a singleton-state computer system by all individual computer systems that are participating in the Ethereum network[24, p.48]. The initial, human-readable code is processed in a isolated to ensure security so that the code from the contract account in question is only able gain access to the wider Ethereum network after it is compiled as bytecode[28, p.33].
The initial code is known as a *smart contract*. Typically the language of choice for a smart contract in *Solidity*, a programming language remarkably close to JavaScript that has been developed by the founders of Ethereum and that is able to compile bytecode for the EVM[22, p.39]. Given the fact that one of Ethereum's main features is the validated transfer of cryptoassets, the essence of smart contracts is the transfer of ETH or other cryptoassets according to predefined conditions. Once conditions are met, the respective contract account executes its programming automatically and once smart contract is live on the Ethereum network, its code can not be altered any longer[24, p.61]. It should be mentioned that funds entrusted to a smart contract are held in escrow and once the contract execution begins, it is impossible to back out or to reverse the transactions[24, p.51]. A smart contract that is connected to a web front-end user-interface and possibly also to a storage or messaging protocol is called a decentralized application also known as *Dapp*[22, p.24].

2.2.4 Transactions and Gas

Keeping in mind Ethereum's essence as a *state machine*, a transaction is a *valid* transition between two states, wherein a transition is formally annotated as[12]:

$\sigma_{t+1} \equiv \Upsilon(\sigma_t, T)$

Herein, T represents the transaction, Υ represents the state transition function and σ represents the respective states before and after the transition[12]. Following section 2.1, it should be noted that both state and computational processing during transition can be arbitrary. For instance, an external account to external account transaction can involve a change of state in

the form of changing account balances of ETH where the quantity of ETH in the sender account decreases and the quantity of ETH in the receiver account increases by the same value. On the other hand, an external account to contract account transaction or a contract account to contract account transaction can trigger the execution of code stored in the *Codehash* field of the receiver account.
As transactions may trigger executable code, the Ethereum blockchain is faced with a fundamental computational problem. The *halting problem* expresses that it is impossible to determine a-priori whether a program will eventually finish its execution or continue to run forever[29]. Accordingly, machines running the EVM could be potentially subjected to a *denial-of-service* attack with a series of transactions that lead to bytecode instructions that are designed to never halt[27]. Eventually all available computational resources will be occupied as the Ethereum network grinds to a halt. Ethereum's answer to this problem is known as *Gas*, a fee paid by the sender account in ETH that is applied to computations performed on the blockchain[28, p.31]. So when a contract account executes its code, each computational steps is charged meaning that DOS-attacks are infeasible on the Ethereum ledger. The price of gas is determined by the market where for each would-be transaction a cost limit for gas is set and where miners choose which transactions they wish to mine into a block resulting ins an equilibrium[27]. Equipped with this information, we can now specify the information included in each transaction[12][30]:

- **nonce**: Number of transactions sent by the sender account; implemented to prevent replay-attacks[22, p.83].
- **gasPrice**: Amount of wei[1] per gas the sender account pays.
- **gasLimit**: Maximum amount of wei per gas the sender is willing pay. The amount is paid upfront and any remaining amount is refunded to the sender account. Should the transaction fail, any gas already spent is not refunded.
- **data**: Arbitrary data field usually used for smart contract calls.
- **to**: 160-bit address of the message call's receiver.
- **value**: Amount of wei to be transferred to the message call's receiver.
- **v,r,s**: signature data of the transaction.

2.2.5 High Level Perspective of Ethereum

From a high-level perspective there are three major building blocks to Ethereum. As illustrated in figure 2.1, Ethereum consists of the blockchain itself, the network of miners and finally the accounts that initiate transactions. It should be noted that miners and accounts are not exclusive entities in the sense that a miner needs to posses an account in order to collect the mining rewards. An external account can call a contract account, create a new contract account or just send a simple transaction, while a contract account can only respond to a transaction and execute according to its programming. Either way, both types of accounts can initiate transactions as specified in section 2.2.4 which are then broadcasted to the Ethereum network via a software client. The broadcasted transaction is a candidate in the sense that it has not yet been validated by the network. A miner that considers the offered gas price to be acceptable then includes the candidate into its block, calculates the applicable rewards and if suitable computes an updated

[1] Wei is the smallest unit Ether can be divided into: $1ETH = 10^{18}wei$

valid state using the EVM[12]. Ultimately one of the mining machines solves the hash function for the next block and its processed transactions are then added to the immutable public ledger.

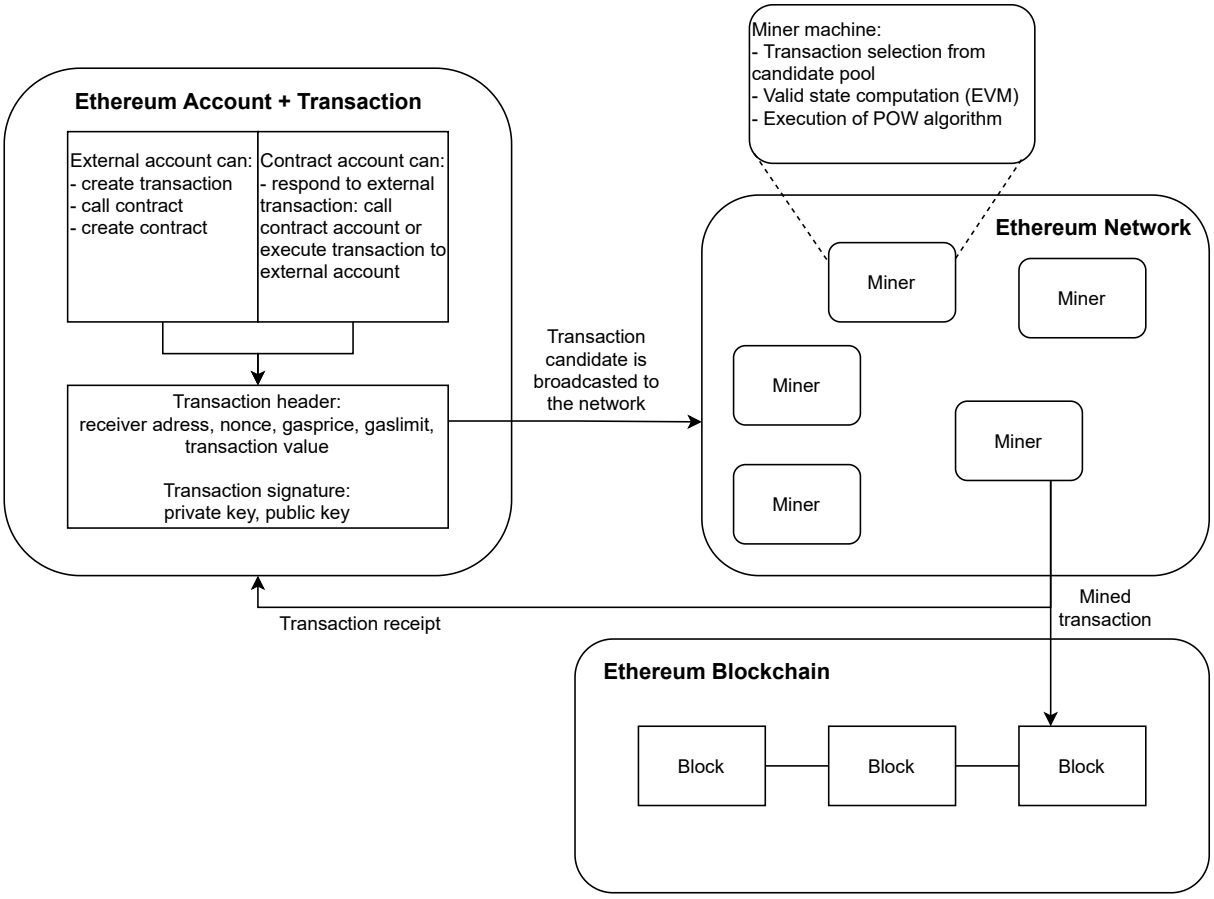

Figure 2.1: High level view of Ethereum [12][28, p.32]

2.3 Cryptoassets

The arbitrary nature of smart contracts and Ethereum's emphasis on transactions favor the deployment of eco-systems to run on Ethereum that show similarities to ordinary monetary systems. Fundamentally money is a system of standardized units for the exchange or transfer of value[31]. Money in the context of Ethereum could thus constitute to standardized units of value that can be transferred between accounts. A striking example is Ethereum's native standardized unit of value *Ether* which among many other use cases is utilized to reward miners or pay for gas[12]. Moreover, the EVM allows for the deployment of smart contracts that specify units of value with distinct characteristics which are transferable on Ethereum[24, p.97]. Based on FATF's definition of *virtual assets*, the following definition will be applied in the course of and for the context of this work[32, p.15][24, p.98]:

Definition 1. *A cryptoasset is a unit of value that is transferable in the context of a distributed public ledger.*

It should be noted that definition 1 does not contain the terms *money* or *standardized* since cryptoassets do not necessarily need to resonate with these terms.

2.3.1 Use Cases

Cryptoassets can, but do not need to mimic monetary systems as the EVM is able to processes arbitrary programming. They should be much rather regarded as *social contracts* between their users[24, p.98]. Insofar cryptoassets on Ethereum include, but are not limited to, the following use cases[22, p.106-107]:

- **Monetary unit**
  A cryptoasset can be a standardized unit for the exchange or transfer of value.
- **Equity**
  Similar to shares in an ordinary publicly traded company, a cryptoasset can represent equity in a digital organization.
- **Voting**
  A unit of a cryptoasset can be used to invoke voting rights or act as a vote.
- **Utility**
  Access to certain services can be dependent on the amount of a certain cryptoasset in possession.
- **Collectible**
  A cryptoasset can be an unique digital item for collection.

2.3.2 Cryptoasset Standards

To address the issues of compatibility of cryptoassets with different smart contracts and dapps across the network, Ethereum introduces a series of standards for the design of cryptoassets that run on its network[28, p.80]. Even though there are ten such standards, as of today only two find wide spread application across the network. Before diving deeper into their specifications, we need to introduce *fungibility*. In the context of economics fungibility refers to the seamless interchangeability of units of money[33]. A one-dollar-bill is *fungible* in the sense that it can be exchanged with another one-dollar-bill and the new bill will exhibit exactly the same characteristics and value. The same logic applies to *fungible* and *non-fungible* cryptoassets. The latter are not interchangeable as they posses unique identifiers which makes them especially interesting for the use case of being collectibles[22, p.107]. Unfortunately, non-fungible cryptoassets are not in the scope of this work and will not be addressed any further.
In Ethereum the standard for fungible cryptoassets is known as *ERC20* and therein is defined a common smart contract interface including mandatory and optional functions as well as attributes[22, p.111]. The functions in question along with short descriptions are illustrated in table 2.1. The next building block of the ERC20 standard are data structures, specifically two data mappings needed to keep track of balances and authorized transfers. The first mapping is an internal table that keeps track of the cryptoasset balances by owner accounts[22, p.112]. The second data structure maps from an owner address to a spender address to a specified amount of the cryptoasset that the spender address is allowed to withdraw from the owner address[22, p.112].

Required?	*Type*	*Name*	*Description*
Yes	Function	**totalSupply**	Total number of currently existing units.
		balanceOf	Provided an account address, returns the unit balance of said account.
		transfer	Provided an account address, transfers an amount of units to said address from the account balance of sender address.
		transferFrom	Provided sender address , receiver address and amount, performs transfer of cryptoasset from sender to receiver.
		approve	Provided receiver address and amount, authorizes execution of transfer from sender (approving address) to receiver.
		allowance	Provided owner address and spender address, returns amount that spender is approved to receive from owner.
	Event	**transfer**	Triggered when a successful transfer is completed.
		approval	Triggered when a successful approval is completed.
No	Function	**name**	Returns name of the cryptoasset in a human readable format.
		symbol	Returns symbol of the cryptoasset in a human readable format.
		decimals	Returns number of decimals used to divide the cryptoasset amounts.

Table 2.1: ERC20 functions and events [22, p.111-112][34]

In regard to the workflows concerned with transfers of ERC20 cryptoassets, two can be identified. The first one is a direct transfer where the sender calls the *transfer* function with the receiver's address as well as a specified amount as arguments and consequently the respective account balances are updated and a transfer event is triggered by the cryptoasset smart contract[22, p.113]. The second workflow is indirect as an owner account in possession of a certain amount of a cryptoasset authorizes another contract account to spend a specified amount of the asset on behalf of the owner account[22, p.113]. To do so, first the owner account calls the *approve* function to authorize the spender account to spend a certain amount of the cryptoasset. Once the spender account is triggered through an inbound transaction to spend a certain amount of the cryptoasset, the *transferFrom* function is called and the spender account transfers funds from the owner account to an receiver account[22, p.113]. A use case for the second workflow could be a cryptoasset exchange operation with a specified exchange rate where a governing smart contract is authorized to spend a number of units of one cryptoasset after it receives a number of units of another cryptoasset.

2.4 Cryptoasset Exchanges

Not even a decade after the mining of the Bitcoin genesis, the market for cryptoassets is filled with thousands of new projects and with millions of new market participants joining every year[35]. Consequently, there is a huge demand for solutions that facilitate the exchange of cryptoassets. Historically, the first solutions were entities with many similarities to traditional, non-crypto financial assets exchanges. Such centralized cryptoasset exchanges facilitate the commerce of cryptoasset by matching buy- and sell-orders from market participants through the utilization of a centralized order book[36].
However, smart contracts and the EVM allow for the implementation of entirely novel solutions that have no precedents in traditional markets. Therein, buyers and sellers are no longer in need of a centralized third party that facilitates and oversees the trading activity[15]. As discussed in section 2.2.3, once a smart contract is live it cannot be altered and will execute a certain sequence of actions automatically depending on the transactions it receives. This means that a cryptoasset exchange implemented with smart contracts won't be necessarily governed by a human entity. Instead it only adheres to its underlying programming and is thus thereby able to facilitate decentralized cryptoasset trading. Such type of exchanges are called *decentralized exchanges*. It should be noted that at the present time Ethereum is not the only distributed public ledger capable of hosting decentralized exchanges as such DEXs exist across multiple ledgers. There are currently three widely adapted approaches in implementing a DEX with various degrees of centralization as illustrated in figure 2.2, but the focus of this work will be the so-called Automated-Market-Makers.

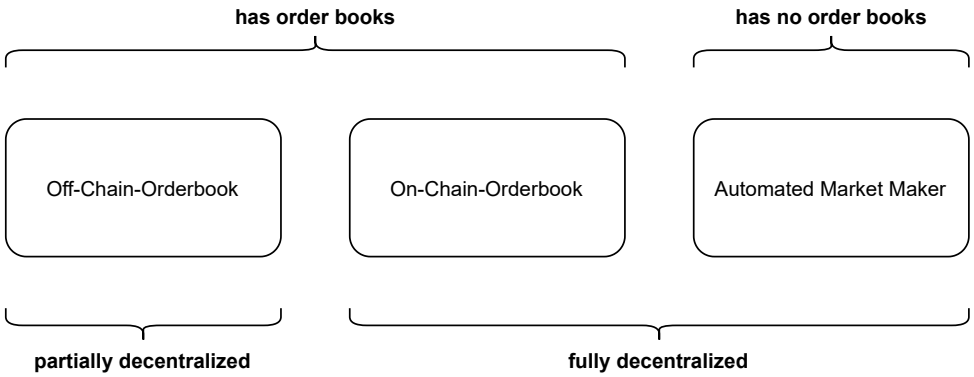

Figure 2.2: Currently adapted DEX types [14][15]

2.4.1 Automated Market Makers

The basic model for *Automated Market Makers* or *AMMs* is much older than DEXs or distributed public ledgers themselves. Already in 2003 the idea of replacing order books with a concept that functions well even on low liquidity is introduced in the context of sport betting markets [37]. In the traditional sense a *market maker* is a market participant that provides liquidity in one or several trading pairs in an order-book based exchange by offering huge quantities of both buy and sell orders while remaining delta-neutral[38]. Besides possible flaws, such as security or trust, that come from the aspect of centralisation of order-book based exchanges, they can also carry additional risks for low liquidity trading pairs. As market makers usually

profit from the bid-ask spread in the order book, they may try to reap additional profits from thin order books by placing their orders in such a way that increases the spread at the expense of other trading pair participants[15]. On the contrary, AMMs neither require an order book nor market making in the traditional sense. An AMM cryptoasset trading pair consists of a liquidity pool for each respective asset and a deterministic pricing formula that sets the exchange rate[14]. Therefore, instead of a direct exchange between buyers and sellers of an asset, there is an indirect relation with the governing smart contract acting as a proxy. Due to the fact that the exchange rate or price is determined algorithmically, as opposed to traditional exchanges where market makers consciously influence the price, AMMs heavily rely on arbitrageurs to equalize the AMM trading pair exchange rate to the market wide exchange rate[15][14]. Since all AMMs work with liquidity pools, the feature that defines different types of AMMs is the pricing formula. A selection of possible AMM implementations is summarized in table 2.2. Herein, exchange rates are derived from the balance between the asset pools. The balances can be calculated through simple summation, as is the case with CSMM, or through more elaborate methods. For a deeper outlook on different approaches to AMM implementation, we refer the reader to [15]. There is, however, one type of AMMs that dominates all other implementations of AMMs and DEXs in the context of trading volume. As such, this work will focus on the most widely adapted AMM, the *Constant Product Market Maker* or *CPMM*.

Name	**Short name**	**Pricing Model**
Constant Product Market Maker	CPMM	$\prod_{i=1}^{2} x_i^{0.5} = k, where \sum_{i=1}^{n} w_i = 1 \, for \, i \in \{1, 2\}$
Constant Mean Market Maker	CMMM	$\prod_{i=1}^{n} x_i^{w_i} = k, where \sum_{i=1}^{n} w_i = 1$
Constant Sum Market Maker	CSMM	$x + y = k$

Table 2.2: Selection of AMM implementation types[15][16][39]

2.4.2 Constant Product Market Makers

Looking at the landscape of DEXs, it is evident that the largest exchanges in terms of trading volume in USD, are all CPMMs. As with all AMMs, a CPMM consists of pooled asset reserves and a governing smart contract. A CPMM trading pair stores pooled reserves of two assets, and provides liquidity for those two assets, maintaining the invariant that the product of the reserves cannot decrease according to the pricing model in table 2.2[16]. The defining feature of a CPMM in regard to the reserve pools is the fact that there are exactly two pools, one for each element of the trading pair[14]. Taking this into account, we can model the trading pair can be as a curve where the increase in one asset immediately results in the decrease of the other asset and vice versa as illustrated in figure 2.3.

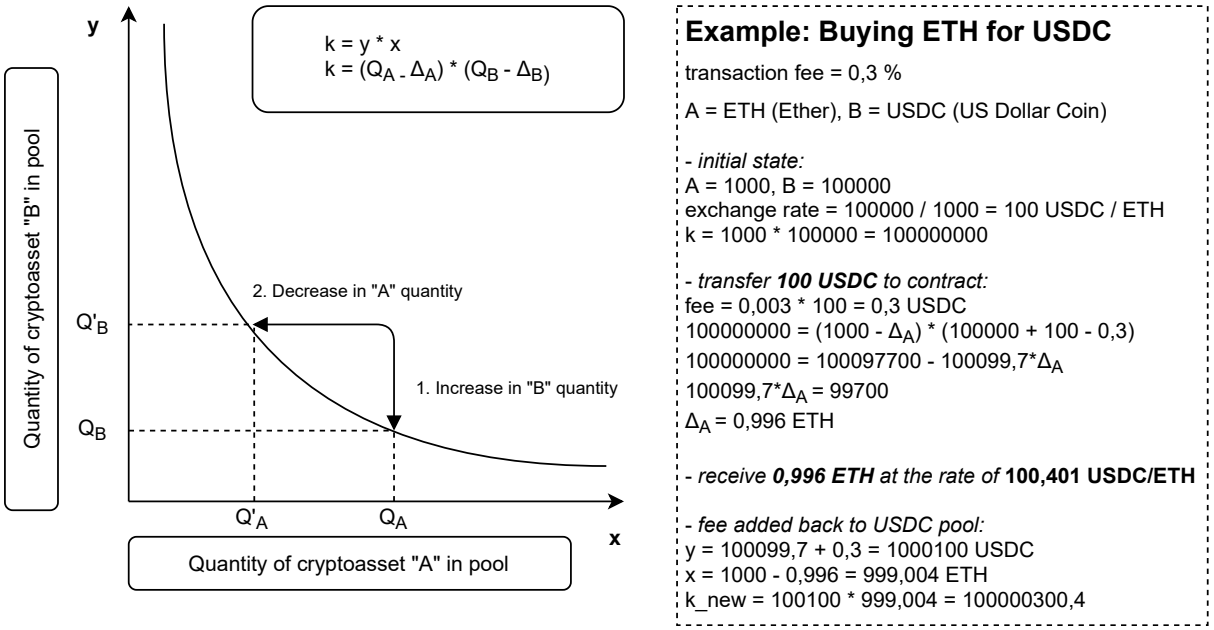

Figure 2.3: CPMM trading pair [14][16]

It might strike as surprising that the example in the beforementioned figure includes a *transaction fee*. Exploring this finding leads to the topic of liquidity provision in CPMMs. The fact that the asset pools need to contain some amounts of assets in order for the trading pair to function should be self-explanatory, but how are these funds provided and for what reason? Liquidity is provided by market participants in possession of the relevant cryptoassets who are required to place an equal amount of both assets into the pools in order to maintain the exchange rate[16][15]. The incentive to become a *LP* lies in the fact that the transaction fee, which is typically below 1%, is paid out to the LPs pro-rata essentially constituting to a passive income[40]. A secondary function of the CPMM transaction fee is to offer a level of price protection to LPs from arbitrageurs[41]. In conclusion, a CPMM trading pair consists of two asset reserve pools and a governing smart contract as the price maker that interacts with external accounts that are either in the role of traders or LPs[15].

2.4.3 Uniswap CPMM

One of the largest DEXs in existence and the experimental environment of this work is Uniswap, an on-chain implementation of an AMM on the Ethereum blockchain using the constant product pricing formula[16][42]. Even though the third major upgrade to Uniswap, the *Uniswap v3*, is already released at the time of the writing of this work, we still focus on *Uniswap v2* since most of the data relevant for this work precedes Uniswap v3[43]. The following section is mainly based on the official V2 documentation and the Uniswap V2 whitepaper[44][16].
Uniswap is a system of smart contracts that can be divided into two parts, one being the *core* contracts which are essential for Uniswap and the other being *peripheral* contracts that support the interaction with the core as illustrated in table 2.3. As Uniswap is permissionless, anyone with access to the Ethereum ledger is theoretically able to list a cryptoasset on the exchange.

Type	**Name**	**Description**	Function	Description	Type
Core	Factory	Initializes trading pair exchange contracts and maps to existing pair contracts.	createPair	Creates pair for two assets if pair does not already exist.	state-altering
			getPair	Retrieves pair of assets if pair exists.	read only
	Pair	Exchange contract (AMM) of trading pair including standard ERC20 functions.	mint	Creates LP token	state-altering
			burn	Destroys LP token	state-altering
			swap	Exchanges two assets	state-altering
	Library	Functionalities to retrieve data and pricing.			
Periphery	Router	Front-end for trading and LP management.	addLiquidity,addLiquidityETH	Adds liquidity to an ERC20-ERC20 pool, Adds liquidity to an ERC20-ETH pool.	state-altering
			removeLiquidity,removeLiquidityETH	Removes liquidity from an ERC20-ERC20 pool, Removes liquidity from an ERC20-ETH pool.	state-altering
			swapExactTokensForTokens, swapExactETHForTokens	Swaps exact amount of input asset for maximum amount of output asset	state-altering

Table 2.3: Uniswap V2 smart contracts with a selection of functions[44][16]

To do so requires an ERC20-compliant cryptoasset account address and an external account in possession of a certain amount of said cryptoasset as well as ETH. The latter is needed for the initial liquidity provision which also sets the initial exchange rate. The *pro-rata* payment of LPs

as specified in section 2.4.2 is facilitated through *liquidity tokens*. An *LP token* is a distinct, pool-specific ERC20 cryptoasset that is *minted* and transferred to the LP whenever someone provides liquidity to a trading pair. When a LP decides to stop providing liquidity, the LP tokens need to be transferred to the exchange contract they originate from. Here they are *burned* and the provided liquidity as well as all applicable rewards are transferred back to the former LP. It should be noted that usually a trading pair on Uniswap consists of an ERC20 cryptoasset pool and an ETH pool. Once all prerequisites are completed, the process of launching a tradeable cryptoasset pair entails the following:

1. *createPair* is called and the cryptoasset account address is passed which in turn emits a *PairCreated* event.
2. *Approve* function as specified in table 2.1 is used so that the exchange contract can withdraw from the external contract.
3. One of the *addLiquidity* functions is called depending on the pool and a certain amount of liquidity is added.

As soon as the above steps are completed, the trading pair is deployed and becomes usable to other market participants. The added liquidity can be removed using one of the *removeLiquidity* functions.

Name	Description	Important information
PairCreated	Emission on successful pair creation in Factory contract	address asset 0, address asset 1, address exchange contract
mint	Emission on successful mint call in Exchange contract	amount asset 0, amount asset 1
burn	Emission on successful burn call in Exchange contract	amount asset 0, amount asset 1, receiver address
swap	Emission on successful swap call in Exchange contract	amount_in asset 0, amount_in asset 1, amount_out asset 0, amount_out asset 1, receiver address
sync	Emission on successful sync function execution	reserve asset 0, reserve asset 1

Table 2.4: Uniswap V2 events with a selection of important information emitted[44][16]

Interacting with an Uniswap exchange contract with the intention to buy or sell a cryptoasset is referred to as *swapping*. There are three possible approaches to swapping, however, one of them has not yet been implemented in smart contracts[45]. The first approach is illustrated in figure 2.4 and constitutes to a direct swap where an amount of *cryptoasset A* is swapped for an amount of *cryptoasset B* as both assets are pooled in the same exchange contract. In the case that a direct swap is not possible, because the particular pool does not exist, an indirect approach needs to be applied. Herein, *cryptoasset A* is first swapped for ETH and ETH is then swapped for *cryptoasset B*[45]. All swap operations should be initiated through the *Router* to ensure safety using *swapExactTokensForTokens*, *swapExactETHForTokens* or similar functions which can be found in the Uniswap V2 documentation[44]. Finally it is important to note that all

state-altering Uniswap V2 functions emit *events* on successful execution as specified in table 2.4. An emitted event can be attributed to a specific exchange contract and contains relevant information about the executed transaction. As these events are recorded on the blockchain, they can be utilized to gain insights about processes happening in the context of the Uniswap smart contract system.
Lastly, we should compare *swapping* with traditional approaches to buying or selling assets on CEXs. As explained in section 2.4, on order book based exchanges the selling and buying of assets is facilitated through what is known as *orders*. An order is a request from a market participants to the entity that governs the order book. Even though many different types of orders exist, the most popular order types are the *market order*, the *limit order* and the *stop order*[46]. A market order requests the immediate execution of the order, thus to either buy or sell an asset at whatever the current price may be. A stop order requests the immediate execution of the order once the price of the asset passes a specific value. This order type is often used as a protection against unexpected price crashes. A limit order is a request to either buy or sell an asset at a specific price meaning that the order is unlikely to be executed immediately. Most CEXs maintain a specialized limit order book where all limit orders are stored so that limit buy and limit sell orders can be matched efficiently. [47] shows that limit order books and constant product market makers are closely related approaches to the design of markets. However, a key difference between a CPMM and a limit order book based exchange is the fact that liquidity providers on the former are bound by the pricing curve explained in section 2.4.2. In practice this means that a *swap* on a CPMM constitutes in practical terms to a *market order* since the swap is executed immediately at a price that is dictated by the pricing curve as illustrated in figure 2.3.

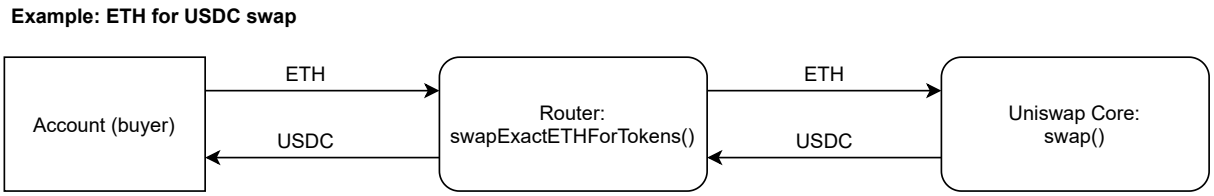

Figure 2.4: Direct swap on Uniswap: Ether for US Dollar Coin[45]

3 Related Work

The context of this work are malicious or fraudulent practices on the Ethereum public ledger and the means to produce results is the analysis of the information available on the blockchain. As such this section consists of two parts. First, we provide an in-depth overview over the state of the art in *on-chain* analysis, followed by an examination of existing research about malicious practices on the Ethereum Ledger.

3.1 On-Chain Analysis

Somin et. al. show that ERC20 token transaction data exhibits characteristics of human social networks by analysing transactions between the years 2016 and 2018[48]. In order to describe the network's structure they apply typical elements form *graph analysis* such as the degree distribution. Herein, the authors refer to the characteristic that a random or dynamic network usually exhibits a *Poisson* degree distribution, while social networks are known to follow a *power-law* distribution. The results show that both buying and selling activity for the combined set of all ERC20 cryptoasset networks appear to follow the power law degree distribution. Based on this finding the authors derive several consequences. First of all, the long tail end distribution supports the notion of decentralization in ERC20 cryptoasset networks which is also strengthens the case of network robustness. Secondly, the findings indicate that ERC20 cryptoassets exhibit economic maturity at the time of the writing of their work.
Victor and Lüders provide common characteristics of ERC20 cryptoasset networks that consist from accounts which engage with a common cryptoasset[34]. Similar to [48], the authors utilize degree distributions to characterize the networks. Albeit, here the focus lies on individual ERC20 networks instead of the combination of all networks. Insofar, they are able to show that most of the individual cryptoasset networks are unlikely to follow a power degree distribution. Moreover, those that do follow a power degree distribution do not exhibit the typical organic growth of popular nodes in social networks as popular nodes in ERC20 networks most likely represent DEXs. Besides degree distribution, Victor and Lüders find that network density can provide insights on how a cryptoasset is distributed to the nodes and that the overall clustering coefficient indicates a global *small worlds* ERC20 cryptoassets network. Finally, the results from the examination of the network activity demonstrate that the main use case for most cryptoassets is speculation meaning that assets are only bought to be sold at a later point.
Guo et. al. provide insights about transactions on the Ethereum ledger through the application of the network theory[49]. As [48] and [34], they also apply the concept of degree distribution. Likewise, the initial findings are indicative of a power-law model for the degree distribution of the transactions network. In accordance with [48] they find that a random graph model does not seem be applicable for the Ethereum transaction network. Like [34], the authors find anomalies in the degree distributions and attribute them to the existence of DEXs. Moreover,

the overall topological structure of the transactions network when modeled as a directed graph shows a giant weakly connected component that follows a bow-tie structure. In contrast to [48], the authors argue that a power law distribution is suggestive of a weak network as compromising a few nodes with high degrees is enough to distort the whole network. [49] also investigate the transaction volumes and find that a small number of accounts is responsible for most of the transactions with large monetary volumes, while most transactions can be attributed to only relatively small monetary payments.

Chen et. al. examine transactions data to not only illuminate the holistic network structure, but to also gain insights on major activities happening on Ethereum[50]. Specifically, they provide detailed information on money flow, smart contract creation and smart contract invocation in the form of graphs. The weighted and directed money flow graph (MFG) includes all transfers of Ether across network, the smart contract creation graph (CCG) is a forest consisting of multiple trees with external accounts as roots and the smart contract invocation graph (CIG) is a weighted directed graph with the edges associating caller and callee and the weights indicating the number of calls. Based on those graphs the authors argue that the majority of users do not use the Ethereum network frequently and that users prefer monetary transactions to smart contract invocation which is also supported by the finding that smart contracts are not widely used in general. Specific analysis of the individual activity graphs includes the investigation of degree distributions, coefficients analysis, WCC and SCC analysis, insights on common edges as well as insights on graph evolution over time. Moreover, the authors create a ranking of the most important nodes for each respective graph using *PageRank*. For further details we refer to [50]. Based on the graph analysis a number of applications are proposed. First of all, [50] present a heuristic in the context of attack forensics to find accounts associated with malicious actors. Furthermore, they propose an approach to detect nodes that create abnormally high numbers of smart contracts. Finally, a deanonymization heuristic is introduced that is able to associate account addresses with real identities.

Lee et. al. provide a comprehensive overview over the Ethereum transactions network in general and over its activity related sub-graphs in detail[51]. Specifically, they examine the network as a whole, the sub-graph of smart contract related activities, the sub-graph of transactions in general and the sub-graph of transactions related to standardized cryptoassets. Their findings resonate with [50], [48], [34] and [49]. For instance, the degrees distribution seems to follow a power-law model with a cut-off in most of their graphs. As [50], the authors find that the graphs all contain a strongly connected component surrounded by smaller groups of vertices which is typical for internet based networks. However, as opposed to [48], Lee et. al. argue that Ethereum transaction networks are unlike social networks in several key characteristics.

Extending their previous works that largely model Ethereum transactions as static graphs, Zhao et. al. focus on the temporal properties of the network and its evolution over time[52]. Akin to [51], they examine the evolution of the network as a whole, the evolution of smart contract related transactions, the evolution of all transactions and the evolution of cryptoasset transactions. The resulting analysis indicates an overall fast network growth following a preferential attachment growth model. However, it also shows that the transactions graphs become sparser over time. It appears that the set of external accounts with high distribution degrees remains stable over the years, while the set of high degree contract accounts frequently changes its composition. Finally, the authors develop a heuristic to predict the continuation or survival of communities within the Ethereum network using random forest and logistic regression.

Ferreti and D'Angelo model the Ethereum network as a graph and find the same properties

in the context the graph theory as [48],[49] and [50]. Similar to [52], they compare the graphs from network snapshots which are taken at different points in time. Herein, the authors are also able to confirm that the Ethereum network experienced a strong growth in recent years. They proceed further by linking Ethereum's network growth to the increased popularity of the cryptoasset Ether *ETH* as seen in its USD denoted price surge between the years 2016 and 2018. Finally, the authors provide insights on miner distribution by stating that a small number of miners is mining many magnitudes more blocks than the average miner.
Li et. al. apply topological data analysis (TDA) on Ethereum transactions data in order to predict fiat currency denoted prices of Ethereum based cryptoassets by analysing the resulting geometric properties[53]. Herein, they introduce *Betti-Pivots* which are able to detect anomalous transactions based on observations of persistent topological features of the cryptoasset networks. The authors find that topological co-integration of cryptoassets can carry predictive value towards future co-integration of said cryptoassets prices. As for direct price predictions, satisfactory results are achieved for some cryptoassets, but the underlying transaction data contains too little anomalies to provide a comprehensive conclusion on the usage of TDA for cryptoasset price prediction.
Akin to [34], Chen et. al. provide insights about cryptoasset networks running on Ethereum[54]. To do so, the authors collect transaction data up until July 2019 that is related to ERC20 compliant cryptoassets and find a total of 165.955 fitting assets. However, the majority of those assets appear to be completely inactive based on the measurement of asset transactions. In regard to the origin of the found cryptoassets, Chen et. al. specify that while most assets are created by external accounts, a non-negligible number of cryptoassets are created by contract accounts with the betting market *Augur* being a prime example. As for holders of cryptoassets, a long-tail distribution becomes evident with the majority of holder accounts only in possession of a few distinct assets. The authors suspect that accounts which hold a disproportional high number of assets are likely to be DEXs. Based on descriptions provided on the blockchain explorer *Etherscan*, Lee et. al. provide a taxonomy of cryptoassets identifying 5 distinct asset classes. Finally, the authors inspect the transaction data in the context of trading activity and find evidence which suggests that DEXs may be artificially inflating their trading volumes.
Silva provides an account of Ethereum miner relationships based on the analysis of the transaction network[55]. The underlying data consists of transactions from the years 2015 until 2019 where both sender and receiver are miner accounts. Graphs are constructed for periods of 30 days with miner accounts which mined a block within that period as nodes and transactions between miners as edges. From the distribution of blocks mined per miner, the author concludes that Ethereum mining is highly centralized with the top 10 mining accounts accounting for over 75% of all mined blocks. Furthermore, Silva finds that most of the mining and transaction activity happened within giant weakly connected components in all examined periods. Across giant weakly connected components on average 44% of the hashing power appears to be controlled by miners who are part of strongly connected components within the giant weakly connected components. The author further concludes that the unusually high density of the miner transaction graphs indicates that especially large miners are likely to have relationships between each other that go beyond simply belonging to the same mining pools.
Di Angelo and Salzer analyse Ethereum smart contracts from data that originates in the year 2016 with the goal to identify and characterize cryptoassets[56].Like [56], their heuristic utilizes the abstract binary interface (ABI) that is commonly implemented in smart contracts to provide identification of the individual smart contract functions. Using the ABI, the authors identify

ERC compliant cryptoassets through the contracts' interfaces which are naturally standardized. Additionally, they use emitted events as well as the specific contract names to find partially ERC compliant as well as non-compliant assets. The results show that 98% of ERC compliant cryptoassets satisfy the ERC20 standard followed by cryptoassets that satisfy the ERC721 standard. Other ERC standards are only marginally present. The authors further find that smart contract functions exhibit a low reuse rate across different ERC compliant cryptoassets suggesting a wide variety of cryptoassets usage scenarios. Identification and classification of partially or non ERC compliant cryptoassets proves to be much more of a challenge as such assets do not provide standardized interfaces and can be indistinguishable from non-cryptoasset contracts.
Extending their previous work, Di Angelo and Salzer utilize Ethereum blockchain data to detect cryptoassets that can be deemed as *securities*[57]. A cryptoasset can be regarded as a security, if it serves no other purpose than being an investment instrument that is expected to increase its value in the future. Once again their heuristic utilizes Ethereum's abstract binary interface (ABI). The authors suspect a cryptoasset to be a security if the functions of the underlying contract serve no other purpose than the facilitation of the cryptoasset itself. However, the heuristic lacks off-chain context as a supposed security cryptoasset could have an off-chain *utility*. For instance, it could enable access a certain Dapp or a service. Ultimately, Di Angelo and Salzer find that the majority of cryptoasset contracts on Ethereum exhibit no apparent utility beyond the facilitation of the asset itself.

3.2 Malicious Practices on the Ethereum Ledger

Farrugia et. al. apply supervised machine learning to detect accounts that are engaged in fraudulent or otherwise criminal activity on the Ethereum network[58]. Initially the authors obtain a ground truth data-set from a community driven and community maintained database of accounts flagged for illicit behaviour as a well set of legitimate accounts. They extract features from transactions related to those accounts and derive a total of 42 features. Using *XGBoost* the authors implement a classification model achieving high accuracy and determining the most relevant features for identifying illicit accounts. Ultimately, Farrugia et. al. find that illicit account usually exhibit a lower ETH balance than normal accounts. Moreover, they determine that on average illicit accounts seem to send more ETH per transaction than legitimate accounts. Finally, the authors conclude that the most telling feature is account age as illicit accounts are relatively young which suggests that such accounts are specifically created for an illegal activities.
Akin to [58], Poursafaei et. al. utilize both supervised and unsupervised machine learning to find illicit entities operating on Ethereum[59]. Like the previous work, the authors attain a ground truth data-set from accounts flagged for illicit behaviour by the Ethereum community. However, Poursafaei et. al. also obtain accounts connected to those accounts flagged as illicit. In the following step, they extract a number features including *neighbourhood features* that indicate the relation of illicit accounts to the wider network. The authors then implement the classification using a wide range of classifiers. As the first result they conclude that unsupervised approaches are not suitable for this classification task as illicit accounts are too similar to regular accounts. In the case of supervised machine learning, the authors find that the best results are obtained through *Random Forest*, *AdaBoost* or *Stacking* classifiers.
Chen et. al. investigate the presence of Ponzi schemes on the Ethereum ledger[60]. A Ponzi scheme constitutes to a fraudulent financial operation where instead of a legitimate revenue

stream, earlier investors are compensated through the investments of newer investors. Ponzi or pyramid schemes are on of the most prevalent financial crimes and are outlawed across most of the globe. The money flow thus resembles a pyramid structure as more and more investors are lured into the scam. In the context of Ethereum, the features of a Ponzi scheme need to be encoded into a smart contract. This allows for the detection of Ponzi schemes through the analysis of the source codes of smart contracts. As the basis of the detection heuristic, the authors manually analyse 1382 cryptoassets and find 1251 legitimate contracts as well as 131 Ponzi schemes. From there on they deduct a series of indicative features both from the bytecode of fraudulent contracts and from their transaction histories. For instance, one of those features is the fraction of investors that receive at least one payment from the contract. In regard to the classification approach, the authors opt for the *XGBoost* classifier as does [58]. One of the most telling features turns out to be the presence of a caller function that records the EOAs that interact with the Ponzi scheme. Finally, the authors conclude that before May 2017, 0.15% of smart contracts of the Ethereum ledger are likely to be Ponzi schemes.
Jung et. al. look into Ponzi schemes by implementing a classification model based on a ground-truth data set. As [60], the authors derive a number of features from the bytecode of their ground-truth data set of Ponzi schemes. Likewise, this set of features is expanded by behavioural properties of the fraudulent contracts, thus their transaction histories. Unlike [60], the authors utilize a Gini coefficient to characterize the unequal distribution of funds. Moreover, Jung et. al. choose 3 different classification models, specifically the *J48* decision tree, the *Random Forest* and the *Stochastic Gradient Boost*. Ultimately, they find that code features are far more important for the detection of Ethereum based Ponzi Schemes than behavioural features. Furthermore, the experiments reveals that tree-based classificators perform better than the Stochastic Boost Gradient.
Leaning on the previous works, Bartoletti et. al. also explore Ponzi schemes on the Ethereum ledger[61]. The authors seek to establish common features of Ponzi scheme smart contracts to create an extensive ground-truth data set of Ponzi Schemes in order to measure the impact of such fraudulent operations, Moreover, they look to establish guidelines on how users can protect theme selves against Ponzi schemes. Regarding the first goal, the authors establish 4 key criteria to separate Ponzis from non-Ponzis. Like [62] and [60], the authors manually establish a data-set of 184 smart contracts which they deem to be of fraudulent nature. Based on the ground-truth data set, Bartoletti et. al. create a *Smart Ponzi Scheme* taxonomy in regard to the way the funds are distributed by the fraudulent contracts. For instance, in tree-shaped Ponzi Schemes a new investor indicates an already existing investor as the inviter. The invitors are thus *parent nodes* and receive a fraction of the investments made by their *descendant nodes* incentivising them to lure in as many users as possible. Regarding the economic impact of Ponzi Schemes, Bartoletti et. al. conclude that on average 70% of investing EOAs exhibit a return on investment of zero or lower. The authors further find that some smart Ponzi schemes are not only harmful to investors by their very nature, but also contain hidden vulnerabilities that can cause even more damage. Finally, three recommendations are issued on how users can protect theme selves. Following the first recommendation any investment opportunity that sounds too good to be true, usually is. Secondly, users should analyse the source code of smart contracts before committing funds. Finally, it is also helpful to investigate the transaction logs before executing the investment.

Torres and Steichen illuminate another type of scam on the Ethereum ledger that is known as a *honeypot scam*[63]. Therein users are lured into the trap by a seemingly obvious vulnera-

bility in the code of a fraudulent smart contract (honeypot). Usually a user is incentivized to send funds to the honeypot in order to exploit the contract and make a profit. But once a user deploys funds to the contract, they quickly realize that the honeypot contains other hidden vulnerabilities which lead to a total loss of funds for the user and a profit for the honeypot deployer. The authors identify 24 honeypots from public sources and classify 8 different honeypot techniques. The honeypots can operate on three different levels with the EVM itself being the first, the Solidity compiler being the second and Ethereum blockchain explorers being the third. An example of a honeypot technique is the *type deduction overflow*. Herein the type deduction performed by the Solidity compiler upon reaching a *type var* variable is utilized to mislead the user. The heuristic to identify honeypots is structured into three parts with the first part consisting of the analysis of a smart contract bytecode. Secondly the authors implement a cash-flow analysis of suspicious contracts as all honeypots must be able to receive and transfer funds. As the third step, the authors search for features which are unique to the 8 honeypot techniques. Finally the heuristic is used to investigate 151935 smart contracts from which 460 are identified as honeypots. However, this measure only represents a lower-bound indication as as other, yet undiscovered, implementations of honeypots may exist. Nevertheless, their results make it clear that honeypot scams are not a prevalent issue on the Ethereum network.
Leaning on the previous work, Chen et. al. also implement a detection heuristic for honeypots on the Ethereum blockchain[64]. The authors use the honeypots detected by [63] and 218250 legitimate smart contracts as their underlying data. For feature extraction Chen et. al. opt to focus exclusively on the operational codes extracted from the bytecodes of their samples. For classification, the authors choose a tree-based approach, specifically an adapted version of the Gradient Boosting Decision Tree algorithm called *LightGBM*. Compared to the Gradient Boosting Decision Tree, it allows for a better control of the tree growth and reduces the number of features. To address the imbalance of the data-set, the authors apply under-sampling to the negative samples, thus legitimate contracts. In conclusion Chen et. al. are able to show that identifying honeypots through operational code features is a viable and accurate approach.
Qin et. al. explore the implications of *flash loans* and *flash loan attacks* on the Ethereum ledger[65]. In the context of cryptoassets, a flash loan refers to an operation where the borrower takes out a loan containing one or several cryptoassets and pays the debt plus interest back within the same blockchain transaction. Compared to traditional loans, a flash loan exhibits several novel properties. For instance, there is no default risk for the lender except for smart contract vulnerabilities and there is no need for collateral as the whole process is hard-coded into the smart contract. Flash loans can be used for arbitrage trading, artificial trading volume inflation (wash trading) or collateral swapping where the collateral for a standard loan is swapped using a flash loan. The authors present a number of examples where flash loans are used to exploit smart contract weaknesses and manipulate cryptoasset prices on DEXs resulting in close to risk-free profits for the perpetrators at the expense of other market participants such as liquidity providers. Based on the presented examples, Qin et. al. then develop a theoretical optimization frame-work and show that the executed flash loan attacks are not efficient resulting in opportunity costs for the perpetrators. Ultimately, the question whether all flash loans should be considered malicious by default remains unanswered.
Wu et. al. utilize the properties of the Ethereum ledger to detect phishing scams[66]. Phishing refers to a criminal operation where crucial data, such as log-in credentials that provide access to cryptoassets, is illegally obtained. The perpetrators usually achieve this by the means of a *trojan horse*. For instance, a user could be forwarded to a malicious clone website of a

legitimate DEX where the private keys of the user are logged and later used to drain the victim's EOA. The authors obtain 500 million accounts and 3.8 billion transactions records from the Ethereum ledger, albeit it is not specified which time frame is covered by those records. Moreover, they obtain 1259 confirmed phishing accounts that are marked as such by popular Ethereum blockchain explorers. For feature the authors first utilize extraction a random walk model to find co-occurrences of nodes in close proximity. To find the nodes embedding in the larger network, the authors deem a *Skip Diagram* to be the most fitting solution. Finally, they conduct sampling both in regard to the volume of a transaction as well as to its temporal properties assigning a larger weight more recent transactions. As the underlying data-set is extremely unbalanced, Wu et. al. opt for a SVM based anomaly detection classifier called *Oneclass SVM* in order to identify possible phishing scam accounts. While the overall performance of the resulting system outperforms other classification methods, it does exhibit input parameter sensitivity.
Farrugia et. al. detect illicit Ethereum accounts implementing a supervised machine learning approach[67]. Using a public database that contains accounts marked as illicit by the Ethereum community and a selection of legitimate accounts, the authors obtain a balanced data set of 4681 account in total. Based on the successful transactions executed by the beforementioned accounts, Farrugia et. al. extract a total of 42 features including features such as the ETH balance, the average volume of a received transaction or the number of transactions sent by the account. Like [60], the authors opt for *XGBoost* to solve the emerging binary classification problem. With an average AUC of 0.994 the classifier proves to be highly effective in detecting illicit accounts in a balanced data-set. Additionally, the authors also determine that the usage duration of an account is the most important feature for the classification task as on average illicit accounts exhibit a 70% lower usage duration time compared to legitimate accounts.
Gao et. al. examine the presence of *counterfeit* cryptoassets on the Ethereum ledger[68]. As a decentralized, singleton state machine, Ethereum does not enforce any restrictions on the smart contracts deployment and more specifically on their names. Thus different smart contracts can have the same name and symbol. This is used to create malicious smart contracts that mascaraed as legitimate, well-trusted cryptoassets. The authors implement a rule-based detection heuristic on Ethereum transactional data up until 18 May 2020. As the first step, they collect all smart contracts that carry the same or suspiciously similar names and tickers as the top 100 Ethereum cryptoassets on the blockchain explorer *Etherescan*. The resulting list is cleared of false positives based on a number of rules. For instance, the authors take into account that some cryptoassets might migrate their official account addresses to implement new features or fix bugs. Ultimately, Gao et. al. find 2117 *counterfeit* smart contracts that imitate 94 out of 100 top cryptoassets. Looking at the names and symbols in detail, the results show malicious actors use both identical counterfeits as well as *combo-squatting*, thus creating almost-identical names and symbols. As for the popularity and activity of these malicious smart contracts, the authors finds that both exhibit a long tail characteristic. A small number of counterfeits exhibits thousands of recorded transactions, while the majority has less than 50. Likewise, the life-span for most counterfeits is less than a week, while a few are active for months on end. Analysing the counterfeits deployers, Gao et. al. find that a significant number of creators have created at least two counterfeits cryptoassets, but are responsible for the deployment of dozens counterfeits. In order to identify the use cases for counterfeit cryptoassets, the authors crawl cryptoasset-related web spaces and forums to find posts that mention accounts from their counterfeits-list in connection with scams or other malicious activity. The first major use case is the *air-drop*

scam. Herein, unsuspecting users are lured into sending funds to the counterfeit contract with the promise of receiving a legitimate cryptoasset back at an exchange rate that is far better than the usual market rate. However, what the victim really receives is either a worthless counterfeit or nothing at all. The second use-case are follow-up scams to the first use-case. For instance, after not receiving any airdrop, the victim might be contacted by a fake customer service asking for the victim's private keys is order to resolve the issue. Alternatively, the malicious actor might ask the victim to send even more funds to the counterfeit account stating that this is the only way to revert the initial transaction. In conclusion the authors find that over 7000 EOAs have become victims of scams related to counterfeit cryptoassets and that the perpetrators have gained over 70000 ETH. Finally, the Gao et. al. crawl the web in order to examine to what extent the counterfeit account addresses are advertised online. Unsurprisingly, most of the malicious ads are found on cryptoasset-related forums, Telegram channels and YouTube videos. Due to the relative high degree of anonymity, Telegram is the most used method for advertisement of counterfeit cryptoassets.

Xia et. al. investigate scams that are listed on the Uniswap V2 DEX. To the best of our knowledge [69] is the only published research that covers the same domain as our own analysis. The authors implement the initial data collection with the *Graph Protocol* and collect Uniswap V2 exchange contracts (cryptoasset pools) as well as emitted events in the time frame between 20 May 2020 and 6 December 2020. The data set consists of a total of 25,131 pairs and 20,306,762 events of which over 90% are *swap* events. Moving along, the authors describe the data pointing out that October 2020 exhibits a spike in exchange contracts creation. Furthermore, the activity in the pools measured by the amount of emitted events follows a power law distribution with the top pools being responsible for the vast majority of transactional activity. Naturally, the same power law distribution also applies to the exchange contracts trading volume. Finally, the authors note that over 80% of exchange contracts contain ETH in one of the pools.

Xia et. al. argue that fraudulent exchange contracts exhibit similar features such as relatively short activity periods or names that are eerily similar to the names of legitimate cryptoassets. Therefore the authors propose a machine-learning based classification heuristic to identify scams. As the first step in its implementation, they create a ground truth data-set consisting of 6449 exchange contracts in total. For negative samples, thus legitimate cryptoassets, 2397 exchange contracts are collected which are all listed CEX and are insofar verified in their legitimacy. For positive samples, the authors first flag exchange contracts that exhibit *identical tickers* as the negative samples, but slightly *different names* resulting in 4021 *counterfeit* cryptoassets. Furthermore, Xia et. al. collect 31 exchange contracts flagged as scams on the *Etherscan* blockchain explorer. The positive samples are further expanded by exchange pairs that are created by EOAs that have created other positive samples in the data-set. This *guilt by association* approach yields another 2461 positive samples, albeit the accuracy of this approach is questionable.

Regarding the feature selection, Xia et. al. identify 40 features across four categories. The first category are time-series features, thus for instance the period between the first and last transaction of an exchange pair within the given time period. Another interesting time-feature is concentration of mint and burn events throughout the life-cycle of an exchange contract as fraudulent contracts usually exhibit most mint events in the beginning and most burn events at the end as the perpetrators exit the scam. Further feature categories are transaction features, such as the total number of transactions, and investor features, such as the average number of transactions of the interacting EOAs. Experimenting with different classifiers, the authors find

the *Random Forest* classifier to be most suitable with a precision of 96,45%. Applying the classifier to the wider data-set, 11182 exchange contracts are initially marked as being fraudulent. From this set, the authors pick 3775 cryptoassets that have either similar names to existing, legitimate assets or names that create a false association with a popular insitution, corporation or personality (e.g. *TrumpToken*,*FacebookCoin* etc.). Applying the guilt by association approach to these 3775 cryptoassets, the authors gain additionally 676 supposedly fraudulent cryptoassets. In conclusion, the authors identify 10964 fraudulent cryptoassets with 11269 related exchange contracts both from their initial data collection and from their machine learning efforts.
Analysing the results, Xia et. al. find that that 86% of the fraudulent exchange contracts are drained of their liquidity within 24 hours and 37% are drained within 1 hour. In regard to the *modus operandi* of the scmas, the authors find two overreaching narratives. The most common is the sudden draining of liquidity and consequent sell-off against the remaining liquidity in the pool. The second constitutes to inabilities in the smart contract code itself, such as hidden mint functions. In relation to the creators of fraudulent cryptoassets, Xia et. al. find that 11269 fradulent exchange contracts are created by 6496 EOAs. To further expand the list of scam addresses, the authors mark certain accounts that interact with known, fraudulent EOAs with a total of 41337 *collusion accounts* being identified.

4 Concept and Design

In the course of this section we introduce a definition of the term *rug-pull* based on a series of introductory examples. As the next step, we elaborate on the data needed to verify our assumptions and finally we present our on-chain rug-pull identification heuristic.

4.1 Introductory Examples

A rug-pull is not a specific term that carries a clear and universally valid definition. To the best of our knowledge no authoritative institution has published a definition of a rug-pull as of yet. The term exhibits a cultural background that introduces a high level of ambiguity. It is reasonable to assume that the etymological origins for the expression lie in the popular figure of speech known as *pulling the rug from under* which describes the sudden withdrawal of support from someone or something[70]. In the context of cryptoassets, the term starts to appear with increased frequency during and after the period known as *DeFi summer* that begins the release of the Uniswap V2 DEX in the summer of 2020. Towards October 2020 there is a noticeable spike in the deployment of new Uniswap V2 trading pairs most of which are presumably scams [69]. As the sudden loss of funds to malicious or downright criminal activity on DEXs becomes an increasingly pressing issue, the term rug-pull starts to appear with increased frequency in the public discourse about the issue. Herein, *public discourse* mainly refers to the social media communication between individual users and internet communities with an interest in cryptoassets. Naturally, a decentralized and fractured environment of social media posts, forum entries and anonymous chats leads to a high level of ambiguity in regard to what a rug-pull actually refers to. One could argue that rug-pull is an umbrella term, but then the question arises as for how wide the scope should be. For some it is simply a synonymous for a violent *price crash* of a particular cryptoasset. For others, a rug-pull happens when a project fails to deliver what it promised. However, the expression can also carry narrow constrains turning it into a term that can only be applied to a small number of specific situations. Taking the cultural background and the high level of ambiguity into account, we believe that a *bottom-up* approach is the most suitable in finding an overreaching definition that would satisfy the most of the situations where the term rug-pull is applicable. Henceforth, we now present a number of real-life examples where the term rug-pull is used to describe a series of events. Based on these examples we later derive our definition of a rug-pull.

4.1.1 Unicats (MEOW)

Created as a spin-off of the popular *SushiSwap* DEX, the Unicats projects launches on 20 September 2020 with the ticker *$MEOW*[1]. The unaudited smart contract promises to pay out high *yields*

[1] pool creation transaction hash: 0x730e3a8c3a6e0c137e021438fb442920734d31f36af0f7c56d7c464e7aa60af5

to users that deposit funds to the contract for a certain period of time[71]. Yield is a concept that is similar to *dividends* and in its basic form it constitutes to receiving periodic payouts from a certain amount of cryptoassets that are held in the yield-generating smart contract. Unfortunately for the holders of *$MEOW*, the Uniswap V2 *MEOW-UNI* pool is drained on 22 September 2020, not even 48 hours after contract creation[2]. As explained in 2.4.2, the pool sizes in a CPMM DEX directly influence the exchange rate of two cryptoasset and once one of the pools is drained, the cryptoasset stored in the remaining pool becomes effectively worthless if it isn't listed in any other trading pairs. Consequently, holders that fail to swap their $MEOW assets before September 22 quickly realize that they have become victims of a scam as seen in figure 4.1. The liquidity draining operation on Uniswap and other DEXs delivers a profit of approximately 50.000 USD to the perpetrators[71].

Figure 4.1: Tweet about UniCats $MEOW "rug pull", sourced on 3 October 2021 from https://twitter.com/_-Big_Perm/status/1309185044510945282

4.1.2 Santa DAO (HOHO)

Following in the footsteps of the previous example, a project called *Santa DAO* promises high returns in the form of yield to holders of the *$HOHO* cryptoasset[72]. The main difference lies in the fact that Santa DAO does not have a working product. Instead, the project pretends to implement a financing operation known as a *pre-sale*. Herein a project may offer a certain percentage of the supply of its cryptoassets to risk-inclined investors before the the actual listing of said cryptoasset on a DEX. As the pre-sale rate is usually much lower than the initial exchange rate upon listing, investors can turn a huge profit in a short time by selling their pre-sale allocations once the asset is listed on a DEX. In order to gain traction and attract capital many projects rely on social media influencers that promote the cryptoasset like the first tweet in figure 4.2. Following the timestamps of tweets from figure 4.2 we can infer that the creators of Santa DAO execute their scheme almost immediately after deploying the *HOHO-ETH* pair on Uniswap V2. Deleting their social media channels and website the perpetrators net a total profit of over 100.000 USD both from the pre-sale and the initial trading activity on the trading pair[72].

[2] trasnfer transaction hash: 0x659ada515156a545bd1bde34ca180ecad04bc5a79ede4c7c5820dca56c312fe0

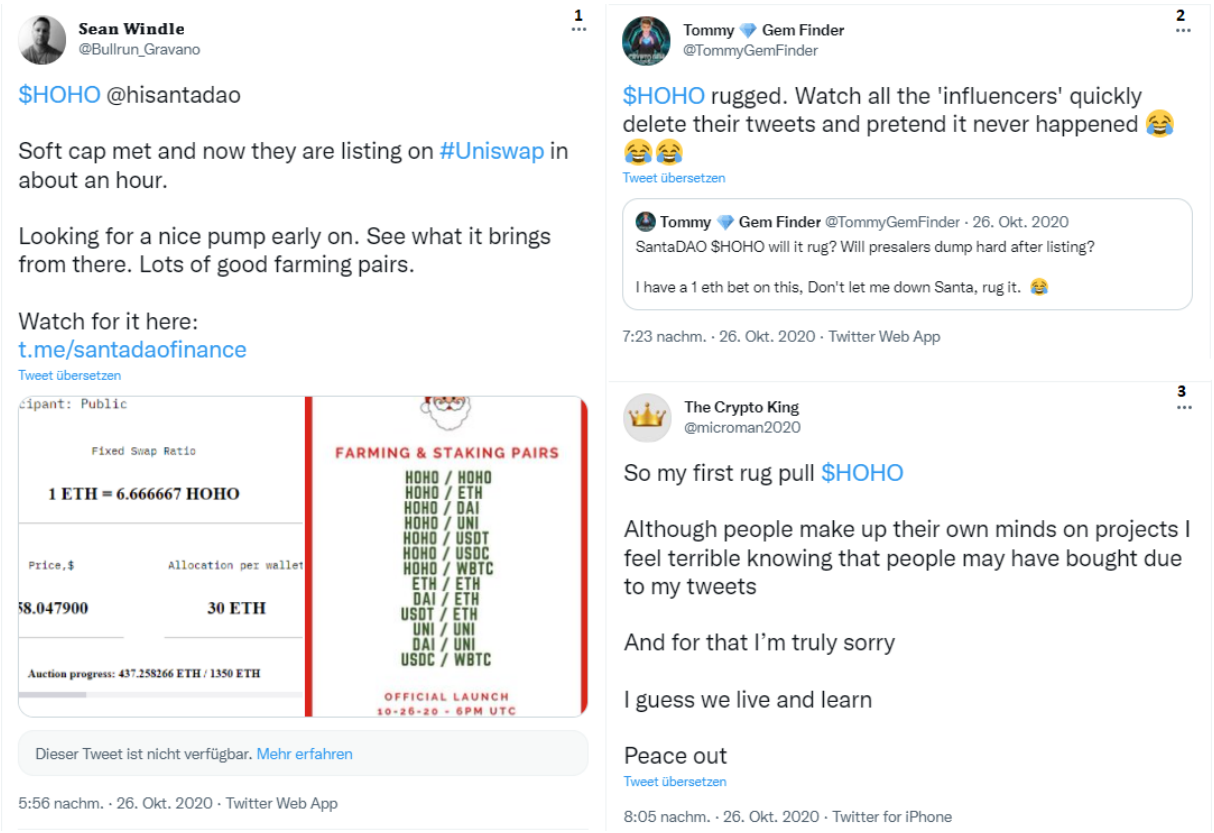

Figure 4.2: Tweets about Santa DAO $HOHO, sourced on 3 October 2021 from (1)https://twitter.com/Bullrun_Gravano/status/1320771284058492929, (2)https://twitter.com/TommyGemFinder/status/1320793087963205634, (3)https://twitter.com/microman2020/status/1320803702182236160

4.1.3 Rexona Finance (RXN)

Not as huge in volume as the other examples, but very much representative of smaller scale criminal schemes that barely leave any trace at all we have the project *Rexona Finance* with the ticker *$RXN*. After a short promotion on Telegram as seen in figure 4.3, the cryptoasset is listed as a *RXN-ETH* pool on Uniswap V2 shortly after 14:30 UTC on 13 January 2021. Approximately 4 hours after listing, the deployers of $RXN activate a function within the contract that allows the trading pair creator EOA to mint an arbitrary amount of $RXN. It should be noted that this function is not hidden or obfuscated within the source code of the smart contract. In theory, anyone knowledgeable in Solidity could predict the outcome after inspecting the fraudulent smart contract. At 19:01 UTC on 13 January 2021 an exceedingly large amount[3] of $RXN is transferred to the RXN-ETH Uniswap trading pair which in effect drains almost the whole ETH pool within a single transaction. As recorded in figure 4.4, the perpetrators net a profit of approximately 18 ETH which amounts to a value of around 20.000 USD at the time of the event.

[3] the amount is larger than the reported total supply of $RXN

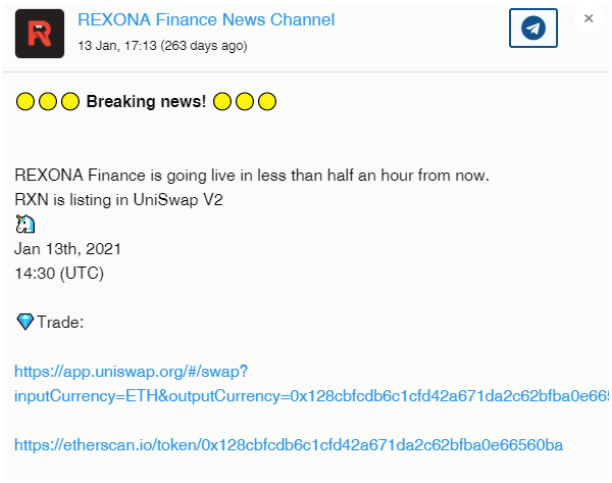

Figure 4.3: Promotional Telegram content about Rexona Finance $RXN, sourced on 3 October 2021 from https://tgstat.com/en/channel/@REXONAfinance/18

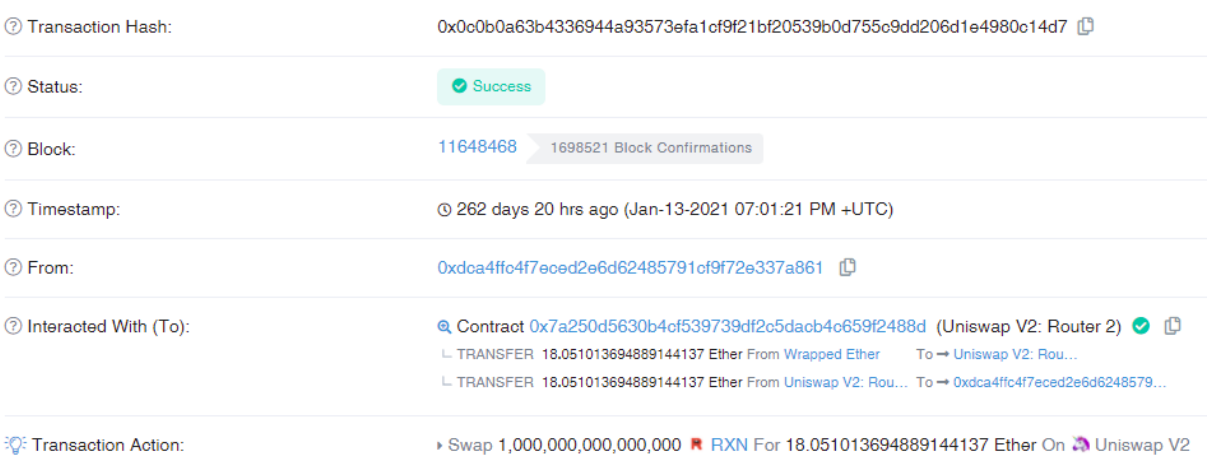

Figure 4.4: Transactional data showing a $RXN to $ETH swap, sourced on 3 October 2021 from https://etherscan.io/tx/0x0c0b0a63b4336944a93573efa1cf9f21bf20539b0d755c9dd206d1e4980c14d7

4.1.4 Hatch DAO (HATCH)

The case of *Hatch DAO* is especially disheartening as investors wholeheartedly believe that their investments are secured by an independent third party service that specializes in safeguarding investor funds against criminal developers and project teams[73]. Following a month long social media campaign, Hatch DAO launches its cryptoasset *$HATCH* at the end of September 2020 on Uniswap V2. The initial liquidity provided by the developers themselves is placed under a time-lock on 19 September 2020 by a third party service which allegedly secures the investors funds as can be seen in figure 4.5(1). Unfortunately, the third party service fails to account for all the possible threat scenarios. As illustrated in figure 4.5(2), on 25 September 2020 the creators of Hatch DAO circumvent the liquidity lock by activating a hidden mint function which results in a *rug-pull* worth over 40.000 USD at the time of the event[73].

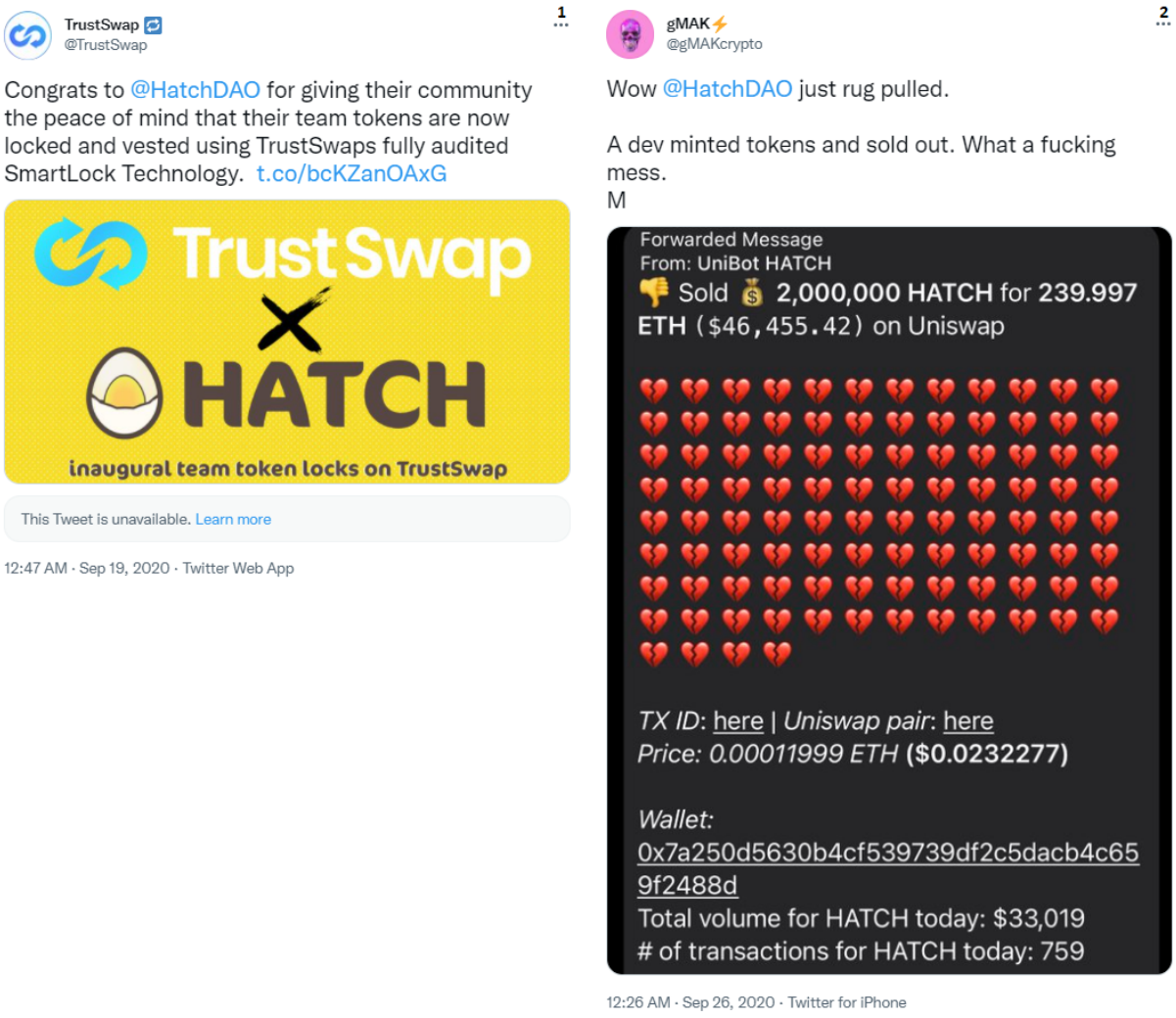

Figure 4.5: Tweets about the allegedly locked liquidity in the HATCH-ETH pool and the subsequent rug-pull, sourced on 6 October 2021 from (1)https://twitter.com/TrustSwap/status/1307088923378757634, (2)https://twitter.com/gMAKcrypto/status/1309620231648800776

4.1.5 Rift Finance (RIFT)

Illustrating the different approaches scammers can use to lure in investors, a project called *Rift Finance* with the ticker *$RIFT* utilizes a form of social engineering instead of creating a false a sense of technical security as in the previous example. Specifically, the creator of $RIFT implements a marketing campaign where they pretend to be a former developer at *Monero*[4] which is an established and highly regarded cryptoasset. In a market that requires technical niche knowledge to understand the products and their implications, a good reputation can generate a lot of goodwill amongst the unsophisticated investors as illustrated in figure 4.6.

Figure 4.6: Positive tweets about $RIFT, sourced on 6 October 2021 from (1)https://twitter.com/DeFiYodda/status/1342467960418557957, (2)https://twitter.com/CryptoWarlordd/status/1341794032993914881

The charade is revealed on social media within a week with users warning the community that the developer of $RIFT is in fact not a former contributor to Monero as illustrated in figure 4.7(1). Ultimately, the victims do not only lose all of their invested funds, but are also publicly taunted by the perpetrators as illustrated in 4.7(2).

[4] https://www.getmonero.org/

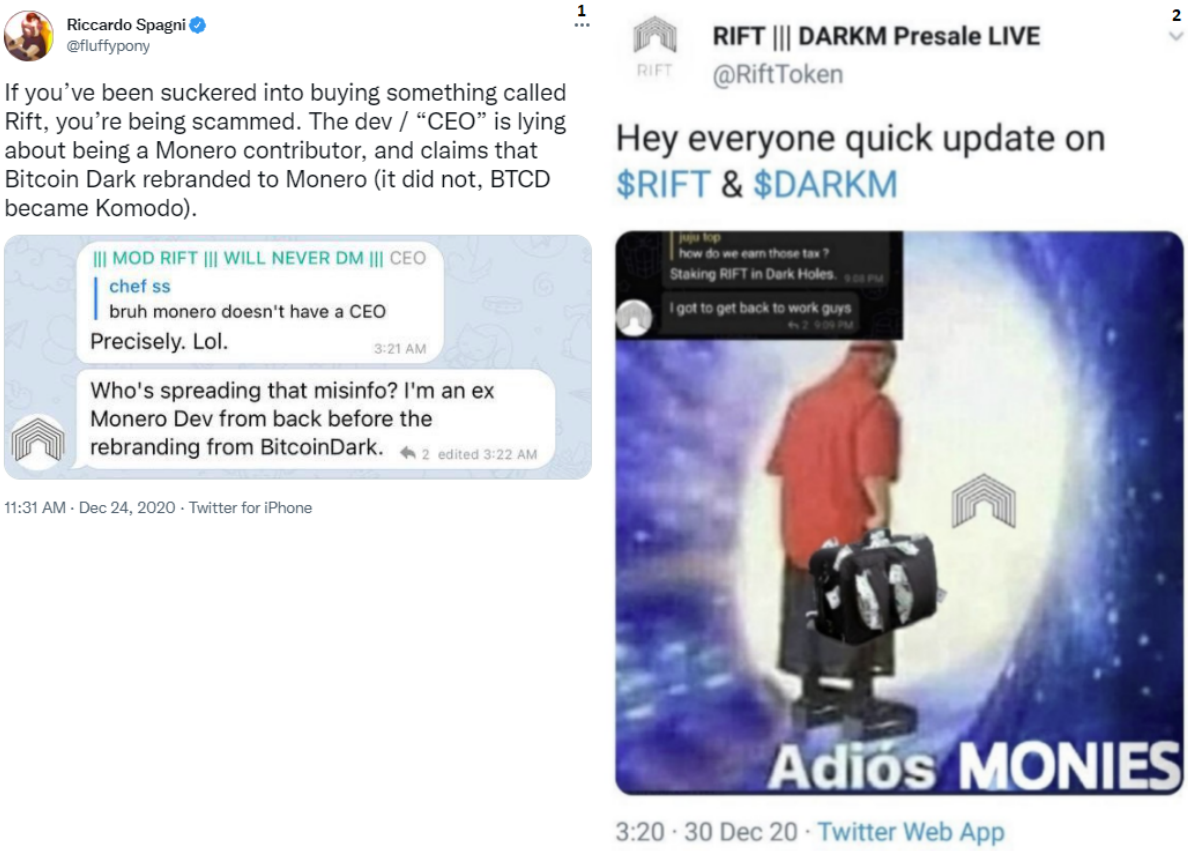

Figure 4.7: Tweets revealing the true face of $RIFT, sourced on 6 October 2021 from (1)https://twitter.com/fluffypony/status/1342055211335036928, (2)https://twitter.com/HaciendoRealid1/status/1344282408825196545/photo/1

4.2 Components of a Rug-Pull from an Investor's Perspective

The examples illustrated in section 4.1 provide a manual introduction to the realities of dealing with cryptoassets on decentralized exchanges. Even though scammers have a vast selection of specific schemes at their behest to extract value from their victims, these kind of criminal enterprises do share a number of common features. On the very basic level *rug-pulls* function like a mouse trap. The unsuspecting victims are typically lured into the scheme through aggressive marketing sometimes combined with an initially extremely well performing asset. Especially the second point can create the so-called *fear of missing out* that often leads to poor judgement calls. Eventually the trap snaps as the perpetrators drain the liquidity pools of the respective trading pair on a CPMM DEX. Even though there are many attack vectors that can facilitate the draining, the execution usually happens within a short time frame so that the holders of the fraudulent cryptoasset have no or very little time to react. Once the scammers have secured their loot, they commonly delete all social media presence as well as apps and websites so that oftentimes the only evidence that remains is the *on-chain* data.

At this point it should be noted that from an investors perspective it does not really matter whether nigh-total loss of funds happens because of clearly criminal activity as illustrated in the examples above or due to other reasons that do not necessarily need to be malicious or intentional. For instance, a critical bug in one or several smart contracts of the project could result in a panic amongst the holders resulting in a massive sell-off. Another possible scenario constitutes to regulatory risk. If the jurisdiction where the legal entity of the project is based suddenly outlaws all cryptoasset-related activities, then all development will come to a grinding halt resulting in a massive sell-off. Especially smaller-scale projects can also be disrupted due to personal reasons such as the death or serious illness of the lead developer.

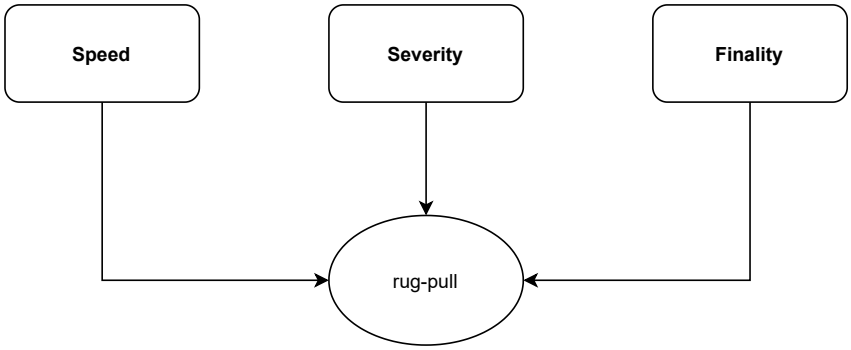

Figure 4.8: Critical criteria for a rug-pull from an investor's perspective

What does matter from an investors perspective is the fast, catastrophic and irrecoverable crash of the value of the asset as illustrated in figure 4.8. The rug-pull needs to be executed relatively quickly so that investors have no time to form suspicions and remove their funds from the contract before or during the rug-pull. Naturally, there are other scams in the field of cryptoassets that see asset prices bleed out slowly over months as the project leaders sell-off their liquidity gradually while soothing the increasingly concerned investors with empty promises and excuses. However, including these operations in the scope of a rug-pull would dilute the meaning of the word up to the point where it simply becomes a synonymous for scam or fraud. The exception to the expectation of speed are trading pairs which were not created with malicious intention in the first place. An example can be rug-pulls that occur due to external circumstances as discussed above. The depreciation of the value needs to be catastrophic in the sense that one shouldn't be reasonably able to attribute it to ordinary market fluctuations. The last and most important feature of a rug-pull from an investors perspective is the *definitive* nature of the devaluation. In our opinion, a necessary differentiation for the term rug-pull lies in the understanding that in the aftermath of a rug-pull a significant share of investors needs to exhibit a net-loss from their interactions with the trading pair. For instance, in March 2020 markets across the globe including the cryptoasset market take a nosedive due to a rapidly spreading epidemic[74]. Most cryptoassets experience price crashes of over 80% in a relatively short time-frame so that one could classify them as rug-pulls based on the criteria of speed and severity of the crash. However, looking at the months after the black swan event it becomes evident that the assets not only recover their lost value, but actually appreciate in price far beyond their pre-March 2020 levels. Thus investors with a long enough time frame have not only recouped their unrealized losses, but are now in profit on their pre-March 2020

cryptoasset investments which makes it very hard to argue for the case of a rug-pull. On the other hand, investors aiming at a short time frame of days or weeks are most likely forced to realize their losses which makes a case for a rug-pull in their context reasonable. In conclusion, a rug-pull can only be a rug-pull if a certain percentage of investors display a net loss after the fact within a given time frame. For the majority of cryptoassets that are publicly referred to as rug-pulls the given time-frame does not matter as liquidity and price never recover given that such fraudulent projects are usually abandoned by both investors and developers.

4.3 Components of a Rug-Pull from an on-chain Perspective

Now that we have established a high-level understanding of the most important rug-pull characteristics, we can examine how these features manifest in the on-chain data. For a given exchange pair all three characteristics can be observed in the *the development of the liquidity pool sizes over time*. Specifically, we are interested in the pool of the *base* cryptoasset of the exchange pair. The term *base* refers here to a set of established cryptoassets that are typically used to fill one of the pools in a CPMM DEX trading pair. As [69] find, over 80% of the exchange pairs on Uniswap V2 have WETH[5] in one of the pools. At this point it should be noted that we focus on Uniswap V2 exchange pairs with WETH as the base cryptoasset, but our logic is applicable to other base assets too. By observing the ebb and flow of the WETH pool size we can find periods where liquidity is drained and consequently measure the severity of the draining. As explained in section 2.4.2, an empty base cryptoasset pool means that the asset in the other pool is effectively worthless on that particular CPMM DEX. Given enough data points, we can also find out whether the liquidity draining is final in the sense that the exchange pair does not recover its lost liquidity within the recorded time frame. However, before going into details we need to discuss the sourcing of the on-chain data. Since the scope of this work is the Uniswap V2 DEX, this discussion will remain a short one, because all the needed data is emitted by Uniswap functions themselves as explained in section 2.4.3. Specifically, the only type of *emitted event* we require is the *sync event*. Upon the registration of a successful transaction the smart contract of a Uniswap V2 liquidity pool emits a sync event which amongst other data contains the updated pool sizes of both pools. In this context a transaction refers to the addition or removal of funds in either of the pools. Thus if each successful interaction with the pool results in an update about the pool sizes, we can collect those updates for each pool into a series which yields us information about the development of the pool sizes over time.
In figure 4.9 we see the development of the WETH pool size in the HATCH-WETH pool on Uniswap V2. The total amount of WETH in the pool at the time of an emitted sync event is shown on the y-axis. It should be noted that the WETH pool does not begin with zero liquidity, because trading pair creators provide a certain amount of initial liquidity during trading pair deployment as explained in section 2.4.3. The x-axis is populated with sync events that occur within the given time frame meaning that the chart shows the pool size development over sync events and not over time. Regardless, the event referred to as a *rug-pull* is clearly visible to the naked eye. After being caught in a continuous downtrend, the WETH pool size drops to nigh zero within a single transaction as documented in the *9403. sync event* that is emitted on 25

[5] Wrapped Ethereum (WETH) represents the same asset as ETH, but results in lower network fees when it is transacted within the Ethereum network.

September 2020 at 21:38 UTC. This coincides with the information we find on social media. A tweet published on 26 September shows that a single transaction is responsible for the crash as can be seen in figure 4.5. We can infer that the base cryptoasset liquidity draining in the HATCH-WETH pool is both catastrophic and so fast that no holder of HATCH is able to react as the bulk of it happens within a single transaction. It should be mentioned that it is possible if not likely that the perpetrators are already in the process of gradually draining liquidity before executing the final blow.

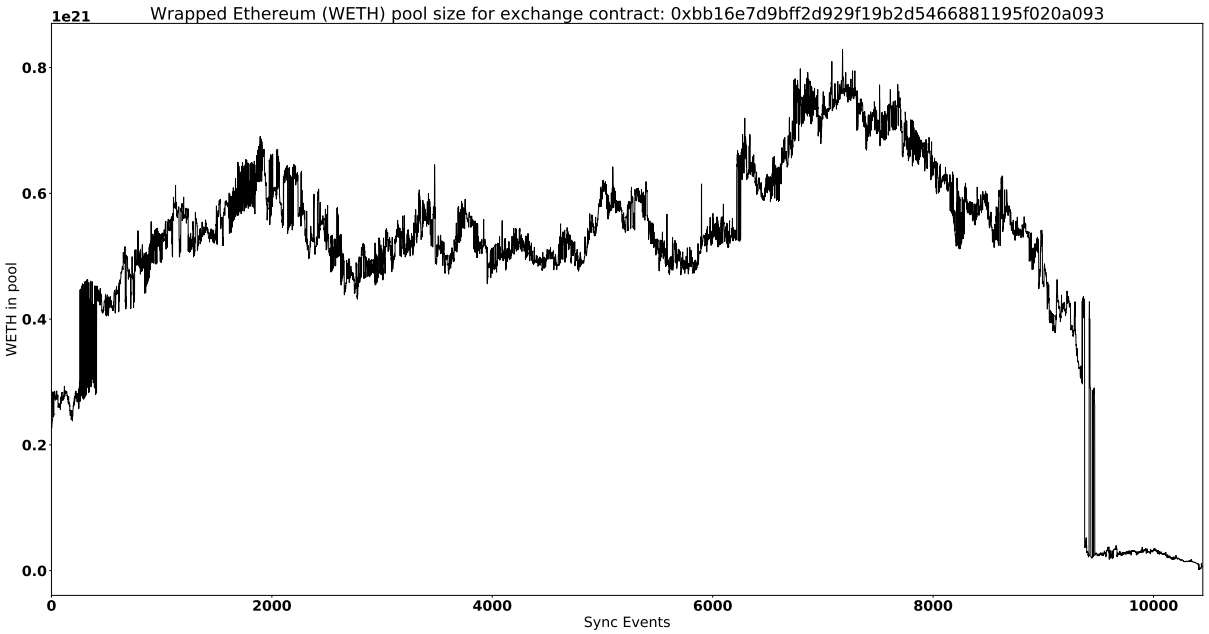

Figure 4.9: WETH pool size development of Uniswap V2 HATCH-WETH pool between first and last sync event within the recorded time frame

Regarding the criteria of *finality*, figure 4.9 illustrates how the liquidity of HATCH-WETH pool never recovers. Unfortunately, figure 4.9 might lead to the false conclusion that the exchange pair ceases to function shortly after the rug-pull. As explained in section 2.2.3, a smart contract that is successfully deployed to the EVM will only stop running once the Ethereum network ceases to exist meaning that it will remain functional even if it is otherwise abandoned. To counter any false impressions figure 4.10 provides a more nuanced view of the temporal perspective. Despite remaining operational up until the end of the recorded time frame, the Uniswap V2 HATCH-WETH trading pair is effectively *abandoned* after the rug-pull. It is reasonable to assume that the vast majority of HATCH DAO investors exhibit a net loss from their interactions with the trading pair. However, in order to corroborate this assumption we need to analyse the EOAs that interact the pool during the recorded time frame.

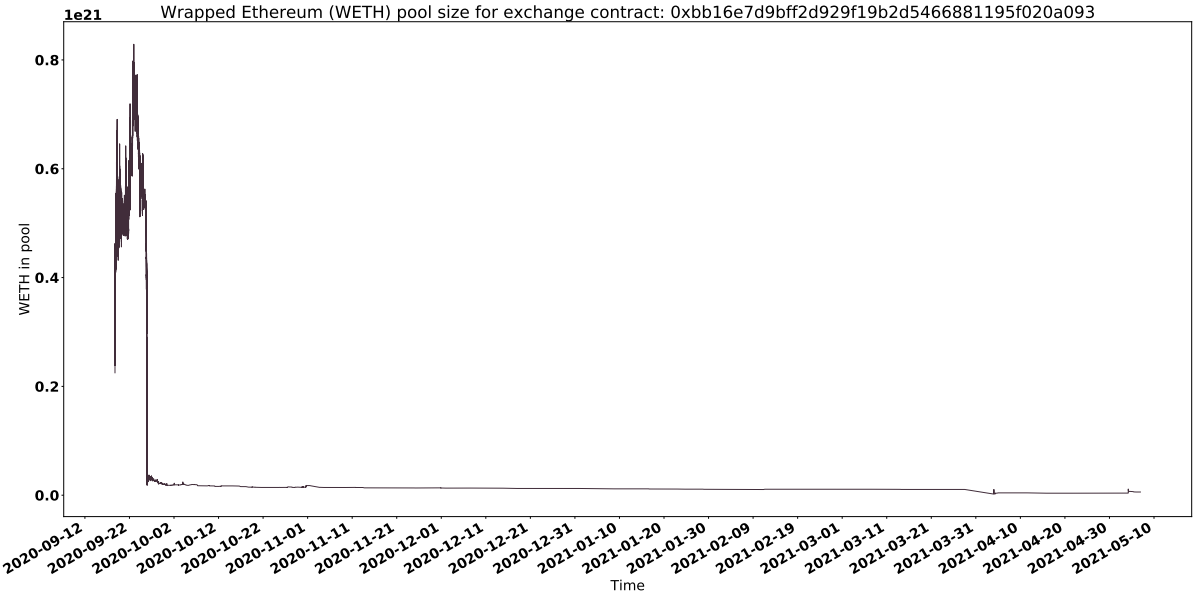

Figure 4.10: WETH pool size development of Uniswap V2 HATCH-WETH pool between September 2020 and May 2021

4.4 A healthy Cryptoasset from an on-chain Perspective

Let us now examine a contrary example, thus a healthy trading pair. The project in question is called *Polkastarter* with the *POLS-WETH* pool on Uniswap V2. It belongs to the category of *launch pads* as it provides incubation services to projects that are looking to launch a cryptoasset[6]. Looking at figure 4.11, it is evident that despite experiencing severe liquidity crashes, none of these crashes happen on a time frame as short as the crash in figure 4.10. For example, between 2 October 2020 and 21 November 2020 the pool size of the WETH pool contracts by over 50%, but it happens gradually including corrective movements to the upside. This means that investors who want to sell POLS without realizing catastrophic losses have a reasonable amount of time and opportunities to do so once a period of dipping liquidity sets in. However, the most important insight from figure 4.11 is the continuous activity throughout the whole recorded time frame indicating that the cryptoasset POLS remains in use. Thus we can conclude that a healthy cryptoasset on a CPMM DEX may experience periods of liquidity contractions, but such contractions should ideally happen gradually and more importantly, a healthy trading pair will not be abandoned even after losing a substantial amount of liquidity.

Figure 4.11: WETH pool size development of Uniswap V2 POLS-WETH pool between October 2020 and May 2021

[6] https://polkastarter.com/, retrieved on 12.10.2021

4.5 Defining a Rug-Pull

Having provided an introduction to rug-pulls both from manual accounts and from an on-chain perspective, we can now move forward to develop a formal definition for the term *rug-pull*. As illustrated in figure 4.8, such a definition should account for the key-characteristics of *severity* and *finality*. We do not use the characteristic of *speed* as under certain circumstances rug-pulls can happen even after significant periods of time. Moreover, the basis for the definition should consist solely of Ethereum on-chain data as it is immutable, can be recovered with relative ease and contains all the necessary information to confirm the key characteristics. This definition is limited to AMM DEXs by the fact that a direct relation between available liquidity and the exchange rate of a cryptoasset is only possible on an AMM. Furthermore, the process of *draining liquidity* as illustrated in section 4.1 cannot be implemented on an order-book based exchange. Nevertheless, we further limit the scope of the definition to only CPMM DEXs, since our analysis does not contain data from other types of AMMs. The subsequent reasoning includes input parameters which we specify based on the analysis of real-world data.
In section 4.3 we explained that a base cryptoasset on a CPMM DEX is an asset from a set of assets that are typically used in one of the pools of the trading pairs. In an effort to reduce ambiguity, we now limit this term to a single cryptoasset. Specifically, we define the **the base cryptoasset** of a CPMM DEX to be **the cryptoasset that is most common across all trading pairs of a CPMM DEX** as explained in definition 2. Following the results of [69], for Uniswap V2 the base cryptoasset is *Wrapped Ethereum (WETH)* as it is present in over 80% of the trading pairs. For a large enough CPMM DEX, defining the base cryptoasset in such a way ensures that it is an established asset that is unlikely to be of fraudulent or otherwise defective nature itself due to the relatively high volume of usage. It is reasonable to assume that critical defects in a base cryptoasset on a large enough DEX would be quickly discovered by the user community. Moreover, it is likely that other cryptoassets on the CPMM DEX are denominated against the base cryptoasset turning it into a measure of worth of other assets. Ultimately, the base cryptoasset pools on a CPMM DEX are also the prices that malicious actors are looking to capture.

Definition 2. *Let S be the set of trading pairs C with $S = C_1, C_2, C_3, \ldots, C_n$ on a CPMM DEX. Let $a \in C$ be a cryptoasset in a pool of a trading pair. Let A be the set of all cryptoassets $a \in S$ with $A = a_1, a_2, a_3, \ldots, a_n$. Let I_a be the set of trading pairs $C \in S$ where $a \in C$ and let $|I_a|$ be the number of elements in I_a. The base cryptoasset a_{base} for a CPMM DEX is the cryptoasset present in most trading pairs $C \in S$ with $|I_{a_{base}}| = max(|I_{a_1}|, |I_{a_2}|, |I_{a_3}|, \ldots, |I_{a_n}|)$.*

Having defined a measure of value on a CPMM DEX, we can now examine the development of value of cryptoassets over time by looking into the development of the base cryptoasset pools in terms of available liquidity. Looking at the WETH pool size development in figure 4.10, it is easy to recognize that the chart can be divided into two sections. The first section exhibits strong volatility and more importantly it also exhibits relatively high levels of liquidity. On the contrary, the second section exhibits very low levels of liquidity that are many magnitudes below the levels of the first section. Naturally, the dividing element between the two sections is the rug-pull event. From this we infer that rug-pulls can be detected *after the fact* by comparing liquidity levels before and after the rug-pull. Consequently, for each transaction we compare the *maximum base cryptoasset pool size* before and after the transaction. Should the ratio fall below a threshold parameter, we add that transaction to a set consisting of suspicious trans-

actions that are likely to be indicative of a rug-pull. As it is possible that the resulting set will contain multiple transactions, the question remains on how to choose the one *most* indicative of a rug-pull. Following the criteria of severity, we choose the transaction with the largest decline in base cryptoasset pool size compared to its preceding transaction. The remaining transaction does not necessarily need to be the transaction that *pulls the rug from under*. In fact, it is unlikely that it will be the transaction that actually constitutes to being the rug-pull, but neither is it our intention to pinpoint the rug-pull transaction. As presented in section 4.1, rug-pulls can be implemented in many different ways. Some can indeed be a single, cataclysmic transaction, while others can consist of a series of smaller transactions. The latter is especially true for perpetrators that try to cover their tracks by sending the captured funds to many different accounts. For this reason we call the one remaining transaction from the set of suspicious transactions a **rug-pull indicating transaction**. Formalized in definition 3, this heuristic is designed not to deliver precise results, but to determine whether a trading pair is subject to catastrophic liquidity implosion and to deliver a good estimate on when this implosion takes place within the recorded time frame.

Definition 3. *Let C be a trading pair on a CPMM DEX. Let T be the set of transactions to C and t a transaction with $T = (t_1, t_2, t_3, \ldots, t_n)$. Let t_{ts} be the timestamp of a transaction $t \in T$ and t_p be the pool size of the base cryptoasset $a_{base} \in C$ at the time of a transaction $t \in T$. Let t_Δ be the base cryptoasset pool size difference between transaction $t \in T$ and the previous transaction $t_{(-1)} \in T$ with $t_\Delta = t_p - t_{p_{(-1)}}$. Let cur be the current transaction $t \in T$. Let T_{bef} be the set of transactions $t \in T$ where $t_{ts} < cur_{ts}$. Let T_{aft} be the set of transactions $t \in T$ where $t_{ts} > cur_{ts}$. Let $mean(T_{bef})$ be the average pool size $t_p \in T_{bef}$. Let $mean(T_{aft})$ be the average pool size $t_p \in T_{aft}$. Let T_{rug} be the set of transactions $t \in T$ where $mean(T_{aft}) < k * mean(T_{bef})$ with $k \in \mathbb{Q} > 0$. Let $i \in T_{rug}$ be the rug-pull indicating transaction with $i_\Delta = max(t_{\Delta_1}, t_{\Delta_2}, \ldots, t_{\Delta_n}) \in T_{rug}$.*

While the heuristic in definition 3 should be able to identify rug-pulls reliably, it is also likely to act overzealous as liquidity implosions can and do occur in otherwise healthy cryptoassets [74]. Accordingly, we need to introduce a safeguard against false positives. To do so we need to get back to a very obvious property of a rug-pull which is the fact that many if not all investors lose their invested funds. In order to determine the fraction of investors that exhibit a negative base cryptoasset balance from their interactions with a trading pair within the recorded time frame, we examine the EOAs that interact with the smart contract. It should be noted that the examination of the EOAs balances only delivers an estimate of the investors balances as it is possible for an investor to own multiple EOAs. We determine the base cryptoasset balance of an EOA by subtracting the total sum of the withdrawn funds from the total sum of the deposited funds withing the recorded time frame. In this regard, *deposited funds* refers to the total amount of the base cryptoasset that an EOA transfers to one pool of the trading pair in order to receive the cryptoasset from the other pool as explained in section 2.4.2. Following the example in section 4.1.4, an EOA can deposit *WETH* in the HATCH-WETH pool in order to receive the cryptoasset *HATCH*. Consequently, *withdrawn funds* refers to the total amount of the base cryptoasset that an EOA receives from the trading pair. We assume that if a trading pair is subject to a rug-pull, then the majority of the EOAs exhibit a negative base cryptoasset balance. We base this assumption on the fact that an nigh-empty base cryptoasset pool will force investors to sell at a loss. Some investors might not even bother to sell at all as the transaction fees would be higher than the returned amount. Both possibilities lead to a negative EOA base cryptoasset balance. Using this heuristic we can determine the fraction of EOAs that exhibit a negative bal-

ance from their interactions with the trading pair. If this fraction exceeds a parameter threshold value, we declare the **EOA balances** of the trading pair to be **indicative of a rug-pull** within the recorded time frame as formalized in definition 4.

Definition 4. *Let C be a trading pair on a CPMM DEX. Let T be the set of transactions to C and t a transaction with $T = (t_1, t_2, t_3, \ldots, t_n)$. Let E be the set of EOAs that execute at least one transaction t to C and e an EOA with $E = (e_1, e_2, e_3, \ldots, e_n)$. Let T_e be the set of transactions t_e to the trading pair C from an EOA e with $T_e = (t_{e_1}, t_{e_2}, t_{e_3}, \ldots, t_{e_n})$. Let $t_{a_{base}}$ be the value of the base cryptoasset a_{base} in a transaction $t \in T$. Let e_{to} be the sum of the values of $t_{a_{base}}$ of all transactions $t_e \in T_e$ where $t_{a_{base}} > 0$ for an EOA e with $e_{to} = \sum_{i=1}^{n} t_{e_{i a_{base}}} > 0$. Let e_{from} be the sum of the values of $t_{a_{base}}$ of all transactions $t_e \in T_e$ where $t_{a_{base}} < 0$ for an EOA e with $e_{from} = \sum_{i=1}^{n} t_{e_{i a_{base}}} < 0$. Let e_{bal} be the base cryptoasset a_{base} balance for an EOA $e \in E$ from transactions $t_e \in T$ to the trading pair C annotated as the difference between the absolute values of e_{from} and e_{to} with $e_{bal} = |e_{from}| - |e_{to}|$. Let E_{neg} be the set of EOAs $e \in E$ where $e_{bal} < 0$. Let $j \in [0, 1]$ be the fraction of EOAs $e \in E$ that exhibit a negative balance e_{bal} with $|E_{neg}| = j * |E|$ and let $h \in [0, 1]$ be a threshold value. If $j > h$, then E_{neg} is indicative of a rug-pull.*

Should the indications presented in definition 3 and definition 4 both hold true for a trading pair, we can then affirm that the trading pair is subject to a rug-pull as formalized in definition 5. It should be noted that this is an *indirect* definition that is based on the consequences of a rug-pull, rather than on the act itself. We choose to apply an indirect definition, because there is great variety of approaches in implementing a rug-pull. A direct definition would be so specific that it only covers a small fraction of events that are referred to as rug-pulls publicly. For instance, if we declare that a rug-pull must happen within a single transaction after a significant mint event, then three out of the five rug-pull examples presented in section 4.1 can no longer be considered as rug-pulls. Of course, a direct definition could be extended to cover all rug-pull implementations known to us, but this would stretch the definition to an impractical length and disregard any implementations not known to us. Besides that, a direct definition would miss rug-pull events that are caused by external factors which are not recorded *on-chain* as explained in section 4.2. However, what we do know is that all rug-pulls on a CPMM DEX lead to a rapid implosion of the base cryptoasset pool size and a loss of funds for the majority of investors. Thus looking for the *on-chain* consequences of a rug-pull, rather than looking for the event itself allows us to cover a broader array of cases.

Definition 5. *Let C be a trading pair on a CPMM DEX. Let T be the set of transactions to C and t a transaction with $T = (t_1, t_2, t_3, \ldots, t_n)$ and let $i \in T$ be the rug-pull indicating transaction. Let E be the set of EOAs that execute at least one transaction to C and e an EOA with $E = (e_1, e_2, e_3, \ldots, e_n)$. Let $j \in [0, 1]$ be the fraction of EOAs $e \in E$ that exhibit a negative base cryptoasset balance and let $h \in [0, 1]$ be a threshold value.* ***trading pair C is subject to a rug-pull if:***

$$j > h \wedge \exists : i \in T$$

Finally, we need to differentiate our definition from the findings of [69]. Xia et. al. refer to rug-pulls as scams where the perpetrators capture investors' funds by draining the liquidity

pools. From a high-level perspective, both definitions overlap in the sense that they both identify the draining of the liquidity pools as the key element of a rug-pull. Unlike us, Xia et. al. do not provide a deterministic heuristic for the *on-chain* detection of trading pairs that are subject to a rug-pull. As summarized in section 3.2, they focus on the detection of fraudulent cryptoassets in general, instead of narrowing down on rug-pulls exclusively. Furthermore, the authors seem to differentiate between rug-pulls and smart contract scams with hidden malicious functions that allow the perpetrators to drain the contract's liquidity. However, from an investor's perspective both actions have the same outcome. In conclusion, we believe that while [69] provide a good general overview over fraudulent smart contracts on Uniswap V2, our work is less ambiguous and exhibits a greater degree of specificity in regard to the subject of *rug-pulls*.

4.6 Ground Truth Data Set

The definition for a rug-pull presented in the previous section is up to this point only based on deductive logical reasoning. We therefore need to validate our reasoning by applying the heuristic on real-world data. Specifically, we need a ground-truth data set with Uniswap V2 trading pairs that have been reportedly subjected to a rug-pull. To the best of our knowledge, no such publicly available ground-truth data set exists at the time of the writing of this work. We therefore must create a ground-truth data set ourselves. In regard to the identification of negative samples, thus healthy cryptoassets, we utilize *CoinGecko* which is a platform that aggregates and publishes data about most of the currently existing cryptoassets[7]. Using CoinGecko we create a selection from the top cryptoassets measured by total market capitalization that have a trading pair on Uniswap V2. In regard to positive samples, the process is more complicated since there is no platform that collects and publishes data about rug-pulls. Nevertheless, as shown in section 4.1, a suitable venue for the identification of rug-pulls without using on-chain data are cryptoasset related social media channels and internet forums. Accordingly we look through the relevant venues presented in table 4.1. It should be noted that it is not feasible to automate this process due to the ambiguity of the subject as the terms *rug*,*rug-pull*, *rugged* etc. are often used to describe anything that is perceived as negative. Simple web scraping for content containing the term rug-pull would result in too many false positives along with huge quantities of irrelevant content. Thus we must conduct our research manually and verify each sample that we find.

Name	**Description**	**Website(s)**
Twitter	Micro blogging platform	www.twitter.com
Telegram	Messaging service	www.telegram.org
Bitcoin Talk	Cryptoasset related forum	www.bitcointalk.org
	Research for relevant news articles	www.theblock.com, www.fullycrypto.com, www.cryptocomes.com etc.

Table 4.1: Relevant online venues for identification of rug-pulls

The most important source of our research by far is Twitter, a micro-blog with the features of

[7] https://www.coingecko.com/en/about, retrieved on 10.10.2021

a social network. This platform is especially well suited for the information we are interested in as the limit of 280 characters per tweet encourages users to write about recent or current events. Once a rug-pull begins to unfold, affected or interested users begin posting about it almost in real-time as illustrated in section 4.1. Moreover, Twitter preserves created content over a long time and includes extensive search functionalities. This allows us to efficiently retrieve tweets about alleged rug-pulls that are months or years old. *Bitcoin Talk* is a forum created specifically to entertain discourse about cryptoassets. This source is provides detailed information on major rug-pulls as users engage in prolonged discussions about the subject. Unfortunately, smaller rug-pulls are seldom mentioned or discussed. The same logic applies to relevant news articles. The largest rug-pulls receive most of the attention from crypto-focused media, while the small ones are not mentioned at all. Finally, we also utilize Telegram in our research which is an instant messaging service that also implements group chats and channels[8] Telegram serves in a supporting role as we look for existing channels and chats about cryptoassets that are allegedly subjects to rug-pulls in order to confirm our suspicions.
Due to the very nature of the source, there can be no absolute guarantee on the truth value of the alleged rug-pulls that we find. Our sources are at best non-peer-reviewed articles based on personal accounts from social media and on-chain investigations where the specifics are unknown to us. At worst the allegations of a rug-pull could originate from the intentional effort by a third party to discredit a certain cryptoasset. Therefore, we refrain from using the term *evidence* and instead search for off-chain *indications* of rug-pulls. However, if we are able to confirm the off-chain indications with our on-chain heuristic, then it is extremely probable that a rug-pull actually happened. Consequently this means that false negatives produced by our heuristic need to be examined especially thoroughly in order to determine whether our heuristic failed or whether the social media allegations are wrong.

4.7 Rug-Pull detecting Algorithms

In order to validate the definitions presented in section 4.5, we need to apply them to the trading pairs from our ground-truth data-set. To do so it is necessary to transform definition 3 and definition 4 into deterministic algorithms that process transactional data emitted by Uniswap V2 smart contracts. This allows us not only to determine whether our approach works from a fundamental point of view, but also enables us to optimize the variable input parameters. In the course of this section we present simplified pseudo-code versions of the algorithms we use. The first sub section illustrates the full implementation of our heuristic, while the second sub section introduces a resource conscious implementation.

4.7.1 Full implementation

Before processing a trading pair in detail, we first need to determine whether that pair is compliant to definition 2 among other parameters. The initial preprocessing presented in algorithm 1 implements two controls. Firstly, we disregard any pair that does not have the Uniswap V2 base cryptoasset *WETH* in either one of its pools. It is of course theoretically possible to apply our heuristic to any CPMM DEX trading pair, but that complicates the process of determining

[8] a channel is a 1:N group chat where the channel creator can publish content, while other channel members are on read-only.

the value of the loss significantly as we cannot be certain whether the captured cryptoasset has value on the market. By focusing exclusively on the base cryptoasset of a CPMM we can circumvent this problem while sacrificing only a small fraction of samples. Secondly, we disregard any trading pair with a low number of transactions. Following [62], we also implement a minimum of 10 transactions per trading pair. The second control measure serves the purpose of excluding trading pairs that are not intended to be functional. As deployed smart contracts can never be deleted, we hereby address the issue of Uniswap V2 trading pairs that are only deployed for testing or experimentation. If a trading pair passes both controls, we also calculate for each transaction the difference in base cryptoasset pool size between the transaction and its predecessor as specified in definition 3.

Algorithm 1 Initial Prepossessing

1: **Input:** Set (S) of Uniswap V2 Sync Events $\{e_1,e_2,e_3,\ldots,e_n\}$ for given time frame grouped by trading pairs $\{C_1,C_2,C_3,\ldots,C_n\}$ where each pair contains a number of Sync Events $\{e_{C_{k_1}},e_{C_{k_2}},e_{C_{k_3}},\ldots,e_{C_{k_n}}\}$
Output: Set of filtered Sync Events
Parameters: Threshold t for minimum number of sync events in C
Definitions: r_0, r_1 = pools of a trading pair C, $WETH$ = Wrapped Ethereum trading pair address
2: **for** e_i *in* S **do**
3: **if** $WETH$ *not in* e_{ir0} or e_{ir1} **then**
4: *Remove* e_i *from* S
5: **if** $\sum_{i=1}^{e_{C_{in}}} e_i \in C_k < t$ **then**
6: *Remove* e_i *from* S
$e_{i_\Delta} = e_{i_{r_{weth}}} - e_{(i-1)_{r_{weth}}}$

Once the initial preprocessing is completed, we identify the rug-pull indicating transaction as presented in definition 3. For each sync event in a compliant trading pair, we create to subsets of sync events. The first subset contains all sync events that occur before the sync event e and the second subset contains all sync event that occur after the transaction e. Thus for each sync event of a trading pair we find two subsets from the whole set of sync events of that trading pair. Using those subsets we identify the average base cryptoasset pool size before the sync event e and after the sync event e. Then we divide the average pool size after e by the average pool size before e. Finally, we compare the resulting fraction against our input parameter p which is a rational threshold value greater than zero. Any sync events of a trading pair that exhibit a pool size relation fraction that is lower than the threshold value p are added to the set of suspicious sync events of that trading pair. As explained in section 4.3, the base cryptoasset pool size after a rug pull is bound to be many magnitudes lower than before the rug pull. Even though we allow p to contain any positive rational numbers, we expect the optimized parameter value to be anywhere between 0.01 and 0.1. The lower boundary of the expected range is not zero, because the logic of a CPMM pricing curve does not allow for a pool to be fully drained. Finally, if the set of suspicious sync events for a trading pair is not empty, we determine the rug-pull indicating transaction by choosing the sync event from the set of sync events with the largest reduction in WETH pool size compared to the pool size reported

by its preceding sync event in the overall set of sync events. We choose the reduction in base cryptoasset pool size as our determining parameter in order adhere to the characteristics of *severity* presented in section 4.2. Choosing the largest reduction in the base cryptoasset pool size compared to the previous sync event will very likely yield a sync event from a period where the base cryptoasset liquidity experienced a fast and significant contraction.

Algorithm 2 Pool Size Relation

Input: Set (C) of Uniswap V2 Sync Events $\{e_1,e_2,e_3,\ldots,e_n\}$ for a trading pair
Output: A single rug-pull indicating Sync Event e_{rug} or $None$
Parameters: Delay d as an unit of time, base cryptoasset pool size relation threshold $p \in \mathbb{Q} > 0$
Definitions: ts = timestamp of a Sync Event, r = baseline cryptoasset pool size
for e_i *in* C **do**
    $e_{ref} = min(e_{i+1_{ts}}, e_{i+2_{ts}}, e_{i+3_{ts}}, \ldots, e_{i+n_{ts}})$ where $e_{ref_{ts}} > e_{i_{ts}} + d$
    $C_{bef} = \{e_1, e_2, e_3, \ldots, e_n\}$ where $e_{i_{ts}} < e_{i_{ts}}$
    $C_{aft} = \{e_1, e_2, e_3, \ldots, e_n\}$ where $e_{i_{ts}} > e_{ref_{ts}}$
    $e_{i_{bef}} = max(e_{1_r}, e_{2_r}, e_{2_r}, \ldots, e_{n_r}) \in C_{bef}$
    $e_{i_{aft}} = max(e_{1_r}, e_{2_r}, e_{2_r}, \ldots, e_{n_r}) \in C_{aft}$
$C_{suspects} = \{e_1, e_2, e_3, \ldots, e_n\}$ where $e_{i_{bef}} / e_{i_{aft}} < p$
if $|C_{suspects}| > 0$ **then**
    $e_{rug} = max(e_{1_\Delta}, e_{2_\Delta}, e_{3_\Delta}, \ldots, e_{n_\Delta}) \in C_{suspects}$

For trading pairs where we are able to identify a rug-pull indicating transaction, we validate our assumptions through the application of algorithm 3. To do so, we first group all transactions of a trading pair by the EOAs that initiate the transactions. Using the difference e_Δ in base cryptoasset pool size between a transaction and its predecessor as specified in algorithm 1, we can calculate the total amount of the base cryptoasset added to the trading pair and the total amount of base cryptoasset removed from the trading pair for each EOA. Consequently, we subtract the total amount added from the total amount removed in order to determine the base cryptoasset balance resulting from the interaction with the trading pair within the recorded time frame for each EOA. Finally, we are able to calculate the fraction of EOAs that exhibit a loss from their interactions with the trading pair. If the resulting fraction is higher than the threshold value t, then we consider the EOA balances to be indicative of a rug-pull. Implementing such a *naive* approach in the EOA analysis obviously leads to diminished accuracy. A savvy investor can perform so well that they exhibit an overall positive balance despite a rug-pull, while a less fortunate investor can incur losses from interacting with a perfectly healthy trading pair. However, due to the pricing curve on a CPMM DEX such as Uniswap V2, all interactions constitute to *market orders* as explained in section 2.4.3. To the best of our knowledge, there are no wide spread adaptions of *limit orders* or *stop orders* for Uniswap V2 at the time of the writing of this work. Consequently, this means that investors that interact with a CPMM DEX trading pair either through swaps or through liquidity provision have no practical way of protecting themselves against unexpected liquidity implosions. The only way they might be able to protect themselves is either through custom made bots or through a continuous manual surveillance of the liquidity levels. In light of this we can reasonably assume that in the aftermath of a rug-pull a significant fraction of EOAs exhibits a negative base cryptoasset balance.

Therefore, we expect the threshold value t to be anywhere between 0.5 and 1.

Algorithm 3 EOA analysis

Input: Set (C) Uniswap V2 Sync Events $\{e_1,e_2,e_3,\ldots,e_n\}$ for a trading pair grouped by EOAs $\{E_1,E_2,E_3,\ldots,E_n\}$
Output: EOA balances rug-pull indication
Parameters: Threshold t for maximum fraction of EOAs with a negative baseline cryptoasset balance within the recorded time frame
Definitions: ts = timestamp of a Sync Event
for E *in* C **do**
    $E_{add} = \{e_1, e_2, e_, \ldots, e_n\} \in E$ where $e_\Delta > 0$
    $E_{rem} = \{e_1, e_2, e_, \ldots, e_n\} \in E$ where $e_\Delta < 0$
    $E_{total\_add} = \sum_{i=1}^{n} e_{i_\Delta} \in E_{add}$
    $E_{total\_rem} = \sum_{i=1}^{n} e_{i_\Delta} \in E_{rem}$
    $E_{balance} = E_{total\_rem} - E_{total\_add}$
$C_{losers} = \{E_1, E_2, E_3, \ldots, E_n\} \in C$ where $E_{balance} < 0$
$C_{frac\_losers} = |C_{losers}|/|C|$
if $C_{frac\_losers} > t$ **then**
    rug_indication = True

4.7.2 Resource conscious implementation

Depending on the amount of different values provided for each input parameter, the number of combinations of different input parameter values might become very high. As our computational resources are not infinite, we need to find a way to limit the processing time without compromising the integrity of our heuristic. For that reason we introduce a re-sampling algorithm that can be applied to the transactions data of a Uniswap V2 trading pair before the initiation of the heuristic presented in algorithm 2 and algorithm 3. We implement a re-sampling process depending on the number of sync events associated with a trading pair. Pairs with less than 2000 sync events are left unchanged. For pairs with at least 2000 sync events, but less than 10000 sync events, we take the first digit of the total number of sync events and set it as the n in every *n-th* sync event we keep in the set. For instance, for a pair with 2545 sync events, we keep every second sync event in the set. For a pair with 9238 sync events, we keep every 9th sync event. For pairs with at least 10000 sync events, but less than 100000 sync events, we use the first two digits of the total number of sync events, but otherwise applying the same logic. Finally, for pairs with more than a million sync events, we use the first four digits of the total number of sync events. Thus, for a trading pair with 7840934 sync events, we keep every 7840th sync event in the set. This heuristic ensures that algorithm 2 and algorithm 3 never process more than 2000 sync events per trading pair if the re-sampling beforehand. We do not expect the re-sampling process to have significant effects on the results of algorithm 2, due to the *finality* of a rug pull. Even if we process only every n-th sync event, the deviation in baseline cryptoasset pool size before and after the rug-pull should still be decisive. On the other hand, we do expect that the results of algorithm 3 can be affected by our re-sampling heuristic.

In order to accurately calculate the balances of the interacting EOAs, we need to process all transactions of a trading pair.

4.8 Parameter Optimization

Our heuristic carries a total of 2 input parameters that require optimization. We implement the optimization process in a staggered approach. Trough the application of our deterministic heuristic on the ground-truth data set, we can extract evaluation metrics that enable us to determine which input parameter combination delivers the best results. First we provide for each parameter an array of initial values that we deem to be reasonable. Once the initial parameter optimization is completed, we initiate a second optimization round. Therein, we discard value ranges that prove to be insignificant or detrimental to the results and increase the granularity in value ranges that appear promising. At this point we need to address the balancing of our ground-truth data set presented in section 4.6 as it is not our intention to curate a balanced set. The reason for this is pragmatical. Negative samples, thus successful Uniswap V2 trading pairs can incur hundreds of thousands of transactions within the recorded time frame, while positive samples may exhibit anywhere between a few dozens to a few thousands of transactions. Given the limited resources of both time and computational power available to us, we deem the limitation of negative samples to be a reasonable trade-off. Even though we have to account for the in-balance of the ground truth data set when evaluating the results, we are able to test larger number of input parameters combinations. As a consequence, we choose the F-Score F_1, thus the harmonic mean between precision p and recall r as the basis for the evaluation. While the F-Score does have its flaws, it is well suited for unbalanced data sets[75].

$$F_1 = 2 * \frac{p * r}{p + r},$$

$$p = \frac{true\ \ positives}{true\ \ positives + false\ \ positives}, \quad r = \frac{true\ \ positives}{true\ \ positives + false\ \ negatives}$$

The specific values we use for each parameter along with short parameter descriptions can be found in table 4.2. For the pool size relation parameter p we start with a pool size relation threshold of 0.01 which means the average base cryptoasset pool size after the rug-pull should never amount to more than 1% of the average base cryptoasset pool size before the rug-pull. The upper range of the values array of p is more of a sanity check of our heuristic than values we expect to be optimal. Regarding input parameter t, we strongly expect the optimal value to be located in the upper range of the values array as the majority of the involved EOAs should exhibit a negative base cryptoasset balance within the recorded time frame.

4.9 Overall Sample

The underlying data for this work consists of events emitted by Uniswap V2 trading pair smart contracts as specified in section 2.4.3. Using the same data collection infrastructure as [34], we obtain Ethereum blockchain data between the May 2020 and May 2021, thus for a period of

symbol	description	values array
p	baseline cryptoasset pool size relation threshold	[0.01,0.02,0.04,0.08,0.16,0.32,0.64]
t	threshold for fraction of EOAs with a negative balance	[0.3,0.4,0.5,0.6,0.7,0.8,0.9]

Table 4.2: Input parameter descriptions

approximately 1 year. This period covers the rise of Uniswap V2 in summer 2020 as well as the subsequent *DeFi Summer* as [69] reports. We can therefore expect to obtain enough Uniswap V2 trading pair samples for our analysis to be viable. A for the specifics of the extraction, we extract swap events as illustrated in table 2.4. Using the pool size parameters of each sync event, we construct the base cryptoasset pool size development within the recorded time frame for each compliant trading pair on Uniswap V2. Ultimately, we apply our optimized heuristic on the overall data-set and evaluate the results. Aside from determining whether a trading pair should be perceived as a rug-pull, we also collect a number of descriptive attributes listed in table 4.3 for each trading pair we process.

Name	Type	Description
pair_created	int	UNIX timestamp of first sync event
is_weth_pair	bool	Indication whether WETH is present in pools
sync_events_count	int	Number of sync events in pair
max_weth_poolsize	float	Maximum WETH pool size
rug_time	int	UNIX timestamp of rug-pull indicating sync event
size_ratio	float	Max WETH pool size after rug-pull as a fraction of max WETH pool size before rug-pull
dur_until_rug	int	Duration in hours between pair_created and rug_time
eoas_count	int	Number of unique EOAs that interact with pair
frac_losers	float	Fraction of EOAs that exhibit a negative WETH balance
is_rugpull	bool	Indication whether pair is subject to a rug-pull
is_resampled	bool	Indication whether pair is resampled
sync_events_count_resampled	float	Number of sync events removed during resampling
resampling_decrease	float	Fraction of sync events removed during resampling

Table 4.3: Attributes recorded during the processing of the overall Uniswap V2 data-set.

5 Data

In the course of this chapter, we present the data we use in the course of our analysis as well as any and all data transformations that are not covered by our core algorithms presented in in section 4.7.

5.0.1 Ground Truth Data-Set

As specified in section 4.6 we manually curate a ground-truth data set to validate and optimize our rug-pull identification heuristic. Ultimately we arrive at a set consisting of 80 samples in total with 45 positive samples ($56,25\%$) and 35 negative samples ($43,75\%$). This means that our data set exhibits a balance of approximately $1,29 : 1$ positive samples per negative sample. The negative samples consist of a selection of successful cryptoassets that can all be found on www.coingecko.com and that have a WETH based trading pair on Uniswap V2. The market capitalization of these assets might rise or fall, but the important point here is that these are all functional and still ongoing projects as can be verified through their official websites which can be found in the attached data. The positives samples consist of Uniswap V2 trading pairs that were subjected to rug-pulls within the recorded time frame. As expected, our most important source for the identification of positive samples is Twitter followed by news articles about rug-pulls found on dedicated websites. The extent of sources for each rug-pull is highly variable sometimes constituting to multiple independent articles and other times to only a single tweet. This means that any false negatives that may arise in the course of the analysis need to be examined carefully as explained in section 4.6. As illustrated in table 5.1, a significant number of malicious cryptoassets carry names that feign a connection to well known assets. For instance, the assets *Yearn Evolved*, *Yearn Recovery*, *YFFS Finance* and *YFOS Finance* all imply a connection to the highly regarded *Yearn Finance* project. This finding conforms with the results of [69] that identify over 4.000 copy-cat cryptoassets on Uniswap V2. Other positive samples try to appear legitimate by including popular terms from the cryptoasset space into their names. A good example in that regard is *Hatch DAO* which uses the popular acronym DAO for *decentralized anonymous organization*. Finally, some of these assets try to directly appeal to simple greed by using terms such as *pump*, *gains* or *only up* in their names.

Name	Ticker	Uniswap V2 Pair Address
Unicats	MEOW	0x39311c70d59866eb18eec2631e5eceddda20f528
Burn Vault Finance	BVF	0x8f017EB62b85D24210f9b14dAdB5dcEC6164625F
Sushi Swap	SUSHI	0xCE84867c3c02B05dc570d0135103d3fB9CC19433
Santa DAO	HOHO	0x3C820C9d611C17fe423D80C3c1e142556CBBB2Fb
Hatch DAO	HATCH	0xbB16e7D9BfF2d929f19b2d5466881195F020a093
Biofarm	BIO	0xc1F5a813D01B9D2fF30FB281A1a12e58B9b9A958
Solid Protocol	SOLID	0x31825230Ee1822bCA7bE4Dc3e01A58ac3EfB7539
Hot Dog Finance	HOTDOG	0xC9e5E923Df73cAa96e6Cba4cB9a16574610Ccd06
Pizza Finance	PIZZA	0x8386e3E719612f1Baaa91cCec46593056594656b
Yearn Evolved	YEVO	0xa9027c6ef402a628d968E961Fd137C6C233a68a1
Tweeba	TWEE	0x1a2Fca4EbB7325cf42545aff08ba92C10fE53f9F
Yearn Recovery	YFIR	0x1dDD1E05F823E39e5e4bF07Dc2b800842c7039cC
Xbase Finance	XBASE	0xC7aBF65e7742aB594722864b53EA6ee7f152CcCc
Wolf Protocol	WFLP	0x7547c5f2A9f4aA33C4D6D6152483662e6c627961
Pump Farm	PPF	0x7422c49c99C8764A560A04e8eA4799E9be92A6D2
Fecore Finance	FECORE	0xfb8D6DB914A636a4437e1056e1780Ed3Ae3AAD5A
Rexona Finance	RXN	0x79Fe076Bf3a9f94d8D5E15460F1E0fEfDA122fDa
YFFS Finace	YFFS	0x1969729b2107B1D56053388d3744eF642589601c
YFos Finance	YFOS	0xfFBe70f5224ec77421edB59cb095248b40d5346b
IBase Finance	IBASE	0x9c8618a8Ecf9096b7025B34Cf7c871a086f2bC08
Defi Base	DEFIB	0x4D3D2bF1DB8815E199aEC9c10523443204770B64
RBase Finance	RBASE	0x5ad0B52264961893e5D9397eC1f22D456ce9f92D
VBase Finance	VBASE	0xe0C1CD323655FDc532eFF2E280c46823f14445d0
Rift Token	RIFT	0x3686FAF4991200Fffb5b832E02Ae15A247c2F234
Dark Matter Token	DARKM	0x90424dEF8cdcD769fE4cC0586E5E59cb0AF374Ec
Gains Farm	GFARM	0x708da4Bc0B6c1e3cA4B1f7bA8eFfB826eaD8E18f
Coin Breeder DAO	BREE	0x1B2C62c36DAA42D9bbD51efB1841455E1bD6E5ed
Coins Swop	CSWOP	0x24717CE1c12F399E57FC3AE1420832eB6aCB17a3
Rotex Finance	RTX	0x8193b625ea60AB8E91094b30FEEA08857dabeE8C
DIPEX	DPX	0x2C105614A0E259F44822B7c3Fb5e3ceB29C076af
Game Swap	GAME	0xe9575d5afda819bf5e20E9A4d52E5cEc076764A7
Kimchi Finance	KIMCHI	0xc4da39e646e7f5d233b89ca0f7b75344e7ddb2cc
Yffi Finance	YFFI	0x6e07a1c50d4db0979a0a65bc8b13744a8a1501de
Swerve Finance	SWRV	0xc139d8450177c0b8c3788608518687b585f7ae5a
Eminence	EMN	0x61d8c3d7ad3c3c00c9e4b8da089e19e57da90b91
Friends with Benefits	FWB	0xddd5d0c02ada9b4c0a1a2b3762ae1200528b3167
Soft Yearn Finance	SYFI	0xb56e8c046cfdd2e5ec7e0be0b1b3c46f58d1f3b6
Axion	AXN	0x63499caacfd364c516075245a2950e830a910b90
Xcore Finance	XCORE	0xbb8a1873c709f3acea0200e98cf7b5536a65002e
P-Ethereum	PETH	0x3bb433ee4f4045d793fb9163e309e6735a7c3df5
Liquidity Income	LBI	0xe5bd741d9f3c861d8c2fd547ff6a19a0d510bd2e
Yen Finance	YFN	0xab664778d218436d508ebc25340ea187687225dc
Zenfuse Trading	ZEFU	0x12b124234dc816ed81d0ecd53ca4838443c41a35
Ypria Finance	yPRIA	0x8a78800b4229b0fa0492fda1553ec76e44a0c637
Only Up	ONLYUP	0x3f0ffa6887e40dfda122a227ba0c910996787c37

Table 5.1: Positive samples from the ground-truth data set

5.0.2 Specifics of Data Processing

The underlying blockchain data for this work consists of events emitted by Uniswap V2 exchange contracts as specified in section 2.4.3. We obtain Ethereum blockchain data between block 10008555 and block 12400000. This interval corresponds to the time frame between 21:07 UTC 05 May 2020 and 11:51 UTC 09 May 2021 covering a period of 368 days and 14,74 hours. Specifically, we extract the events illustrated in table 2.4. The data is stored in a *SQLite*[1] database with each event type being contained in a separate table. However, ultimately we only use the *sync events* in the course of our analysis as they contain all the information our heuristic needs. Detailed descriptions of the individual fields of the stored sync events can be found in table 5.2. As SQLite is not able to hold large integers, the *reserve* fields are stored as string values and need to be transformed into numerical values before they can be processed. Furthermore, each table holds a *contract address* column that serves as an index in order link emitted events to specific exchange contracts. For the chronological arrangement of the sync events we use the *block_timestamp* and the *log_index* fields. Moreover, we need the *from_address* in order to implement the EOA assessment. For querying the database and extracting samples for further analysis, we utilize the open-source *Beekeeper Studio*[2] SQL client.

Name	**Description**
address	Uniswap V2 pair that emits the event
transaction_hash	Unique identifier of the transaction
from_address	EOA that initializes the transaction
to_address	Destination of the transaction
gas	Amount of gas used in transaction
gas_price	Amount limit for gas use set before the transaction
block_number	Number of block where the transaction is stored
block_timestamp	Timestamp of block where the transaction is stored
log_index	Index of transaction within the block it is stored in
reserve_0	Pool size of pool 0 after transaction
reserve_1	Pool size of pool 1 after transaction

Table 5.2: Field descriptions of the sync events table, a bold font means that the specific field is used in the course of the analysis[16]

[1] https://www.sqlite.org/index.html
[2] https://www.beekeeperstudio.io/

6 Evaluation

In the course of this chapter we first present the results of our input parameter optimization. Subsequently, we illustrate the results our heuristic using the optimized parameters.

6.1 Parameter Optimization

In this section we present the results from our first and second round of parameter optimization before reporting on the final input parameters that are used in the subsequent analysis. It should be noted that we use the resource conscious implementation of algorithm 2 as specified in section 4.7.2. Generally, we find that similarly to the work of [66], our algorithms exhibit a strong sensitivity to the input parameters.

6.1.1 Initial Parameter Optimization

As specified in section 4.8, we optimize 2 input-parameters during the initial parameter optimization testing a total of 49 combinations. We present the results for each parameter in the subsequent figures. Herein, the x-axis represents the input parameter value arrays specified in table 4.2. The y-axis represents the *average* f1 score for each input value as each value is a part of multiple parameter combinations. What strikes as surprising is the optimal values range for the fraction of losing EOAs parameter t as illustrated in figure 6.1. Contrary to our expectations, the promising input values are located at the lower part of the range instead of at the upper part. The average f1 score enters a steep downwards slope at the negative EOA balance threshold value of 0,5. As we arrive at the threshold value of 0,6 which means that at least 60% of EOAs must exhibit a negative WETH balance, the average f1 score drops by approximately 50% compared to the previous value. Consequently, the promising value range for input parameter t is located before the threshold value of 0,5. We will therefore increase the granularity of the input values for that range in the course of the second round of parameter optimization. Regarding the third input parameter our expectations were fulfilled. The range of promising values for the pool shrinkage threshold p is located below 0,1 and above 0,01 as can be clearly seen in figure 6.2. We will therefore use this range for the second round of parameter optimization.

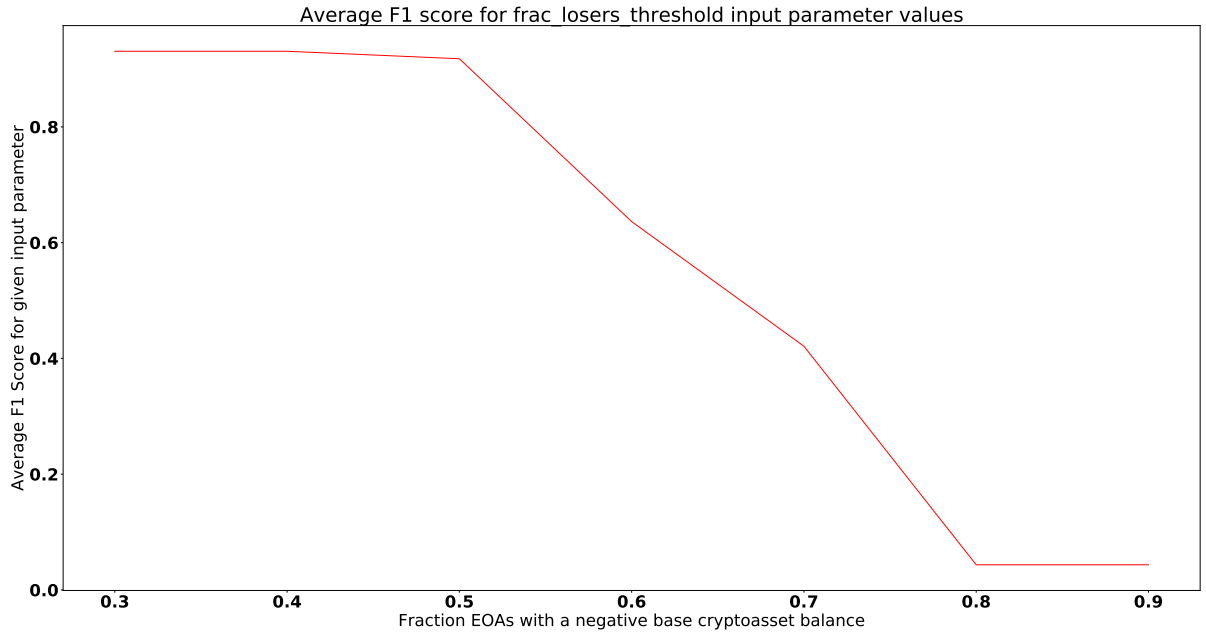

Figure 6.1: Average f1 scores for parameter t values in the initial parameter optimization

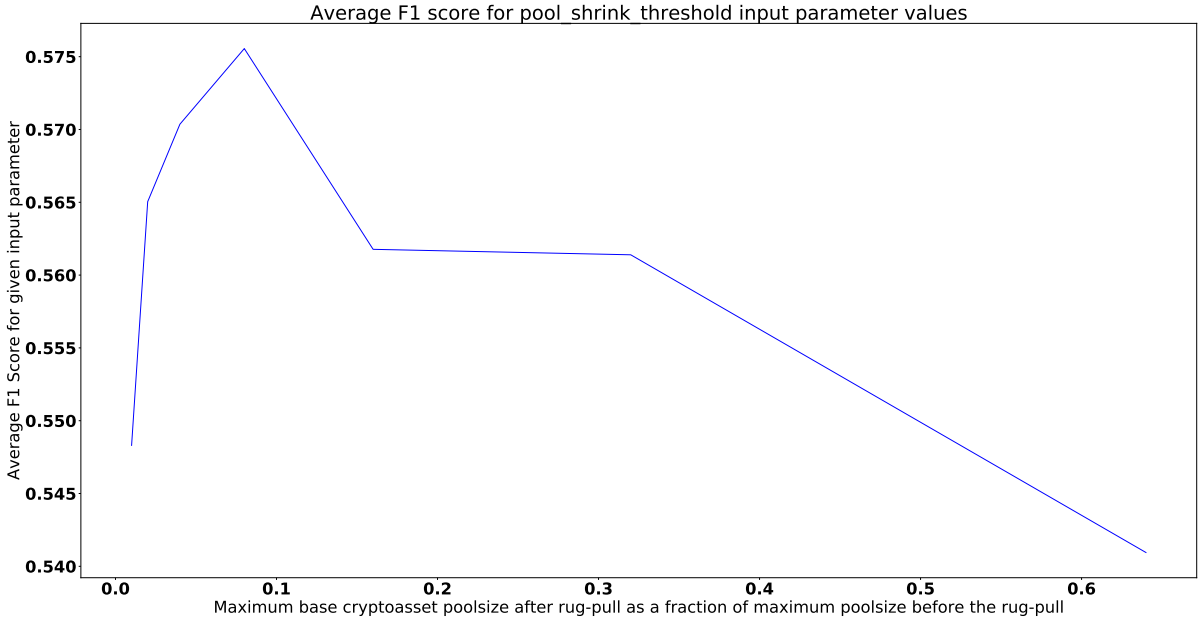

Figure 6.2: Average f1 scores for parameter p values in the initial parameter optimization

6.1.2 Specified Parameter Optimization

As explained in the section 4.8, the second round of parameter optimization aims to optimize input parameters within value ranges that proved to be promising during the initial parameter optimization. Using the same experimental set-up as before, we now utilize the values

specified in table 6.1 testing a total of 399 combinations. As illustrated in figure 6.4, parameter p exhibits the best f1 score for the values of 0,06 and higer. Shown in figure 6.3, the f1-scores for parameter t begin to decline after the threshold surpasses the optimal value of 0,46 which means that at least 46% of interacting EOAs must exhibit a negative baseline cryptoasset balance in order to confirm the rug-pull. The ultimate values we use in the overall rug-pull identification process are displayed in table 6.2.

parameter	**values array**
p	[0.01,0.015,0.02,0.025,0.03,0.035,0.04,0.045,0.05,0.055,0.06,0.065,0.07,0.075,0.08, 0.085,0.09,0.095,0.1]
t	[0.4,0.41,0.42,0.43,0.44,0.45,0.46,0.47,0.48,0.49,0.5,0.51,0.52,0.53,0.54,0.55,0.56, 0.57,0.58,0.59,0.6]

Table 6.1: Input parameter values arrays for the second round of parameter optimization

parameter	*description*	*value*
p	baseline cryptoasset average pool size relation threshold	**0.06**
t	threshold for fraction of of EOAs with a negative baseline cryptoasset balance	**0.46**

Table 6.2: Parameter values of the rug-pull identification heuristic.

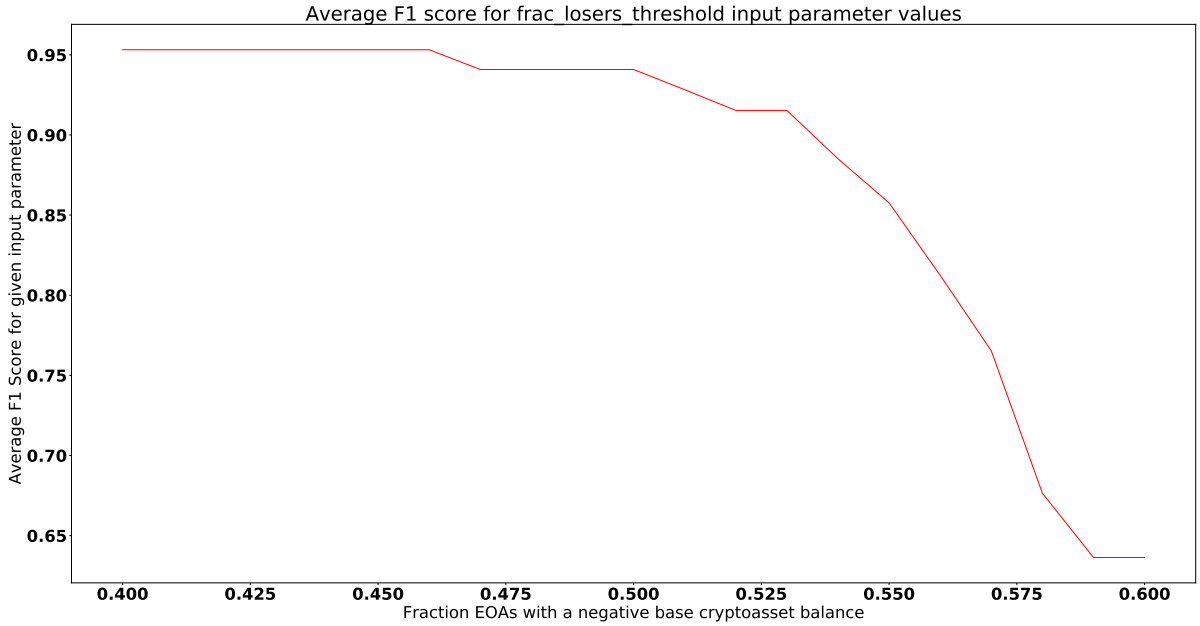

Figure 6.3: Average f1 scores for parameter t values in the specified parameter optimization

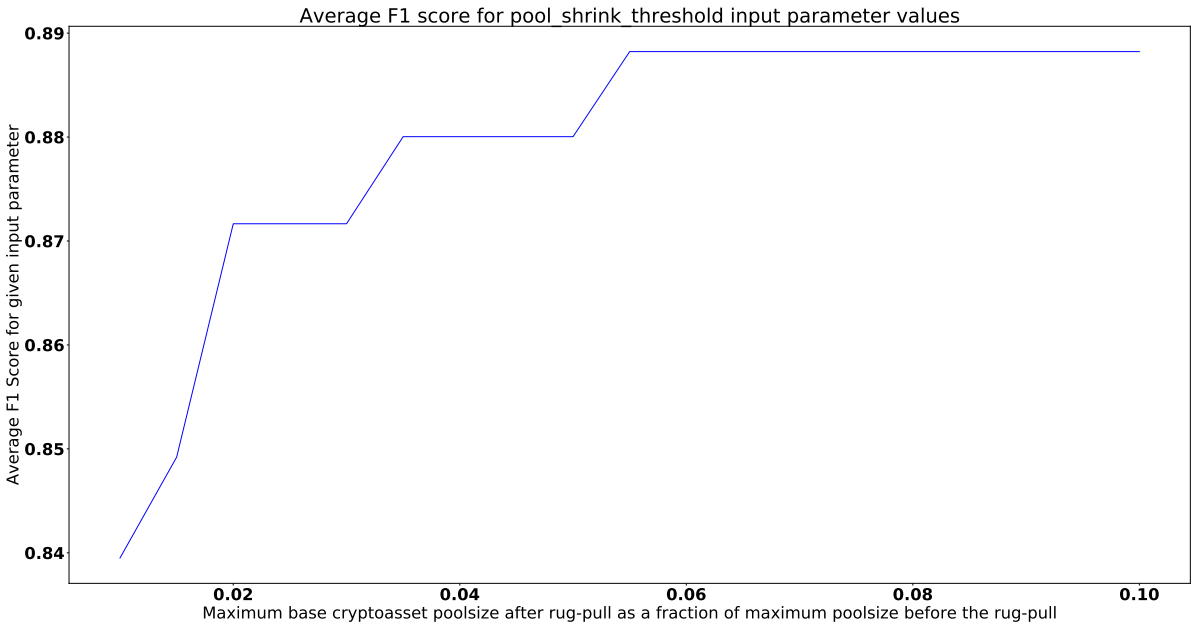

Figure 6.4: Average f1 scores for parameter p values in the specified parameter optimization

6.2 Rug-pull Identification

In this section, we first present a general overview of the Uniswap V2 trading pairs and subsequently we introduce the results of our optimized rug-pull identification heuristic.

6.2.1 General Overview

Our Uniswap V2 data set contains a total of 35.610 trading pairs of which we are able to process all but two. Due to the large number of sync events we are not able to process the *USDC/WÉTH* pair and the *USDT/ETH* pair. However, we can safely attribute that no rug-pull occurred in either one of them as these are the most frequently used stable coin pairs on Uniswap V2. In accordance with [69] we find that 82,42% of the trading pairs have WETH as their baseline cryptoasset. We further find that 53,96% of the trading pairs have 10 or more sync events. All in all the preprocessing presented in algorithm 1 reduces our initial data set by 48,38% leaving us with a total of **18.421 compliant samples**. All results and references presented in the further course of this chapter are based *on the total number of compliant samples* instead of all samples. As illustrated in figure 6.7, the deployment of pairs is not evenly distributed over the recorded time frame. Instead, a massive growth in contract creations per day is followed by a steep decline. This coincides with the findings of [69] that identify October 2020 to be the month with most trading pair deployments up to date.

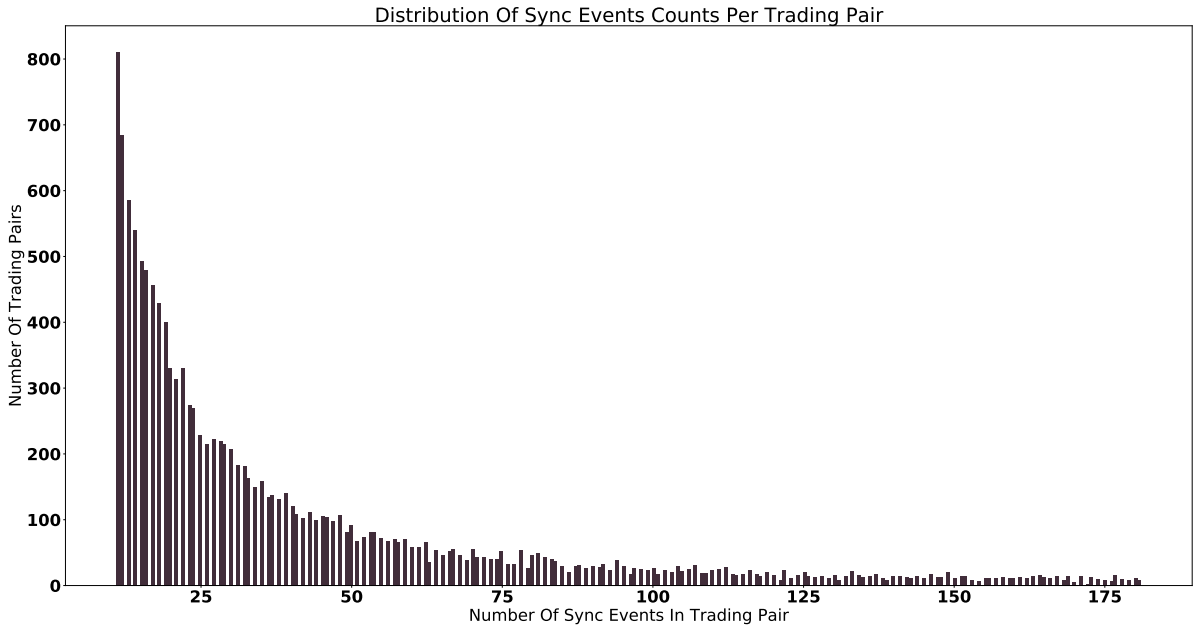

Figure 6.5: Distribution of sync events per trading pair - outlier values excluded.

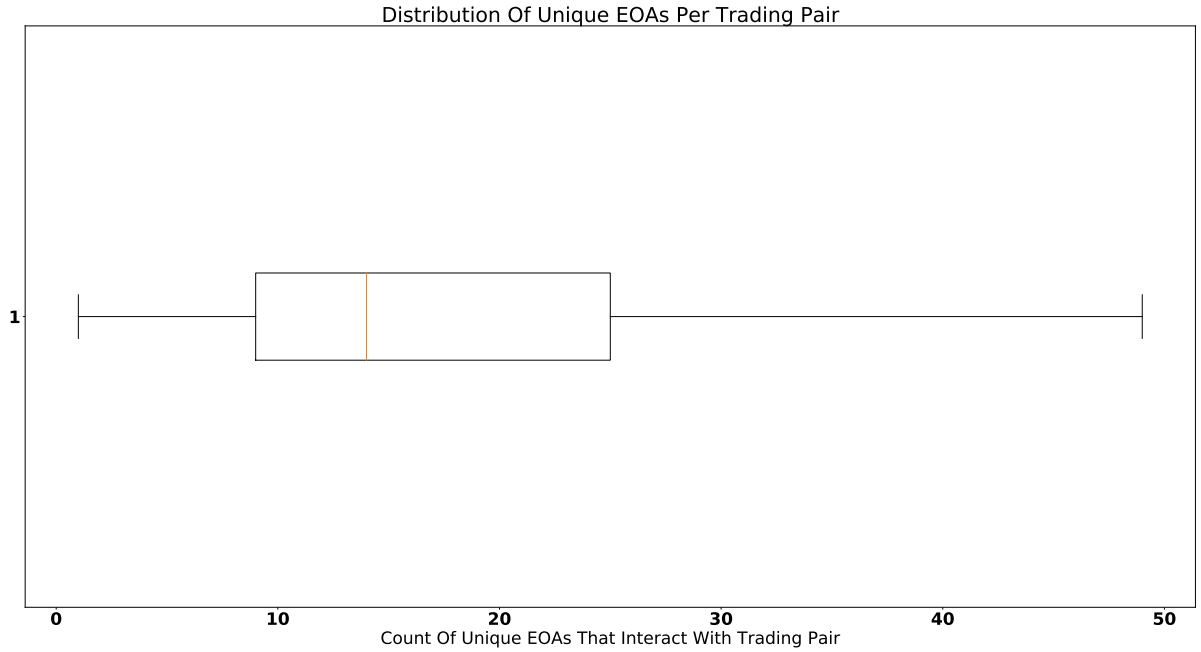

Figure 6.6: Distribution of unique EOAs per trading pair - outlier values excluded.

Looking at the numbers of sync events per trading pair, we can see a power law distribution as illustrated in figure 6.5. It should be noted that the beforementioned figure does not illustrate the full extent of the distribution. Popular trading pairs, especially stable coin pairs like *DAI/WETH* can exhibit well over 1.000.000 sync events. However, the vast majority of trading pairs exhibit a count of less than 50 sync events within the recorded time frame of approxi-

mately one year. This suggests that either most pairs are barely utilized or that a significant number of trading pairs are quickly abandoned after their deployment due to rug-pulls or other factors. In regard to the number of unique EOAs that interact with a trading pair, we also find a long tail distribution. As presented in figure 6.6, over 75% of trading pairs have less than 100 unique EOAs that interact with them. On the other hand, we have a few trading pairs that exhibit unique EOAs counts that are over ten thousand times larger than the median value of 19 interacting EOAs. For instance, the largest EOAs count in our analysis belongs to the *DAI-WETH* pair with 212.969 unique EOAs. In regard to the number of EOAs that exhibit a negative base cryptoasset balance from their interactions with a trading pair, the results show a different picture than the outlook found during input parameter optimization. In the course of section 6.1 we find 0,46 to be an optimal threshold value for the fraction of *losing* EOAs. However, looking at figure 6.8 we can see that in the overwhelming majority of samples at least 46% of EOAs exhibit a negative cryptoasset balance. In fact, the mean value of losing EOAs amounts to 64,33% with a median value of 62,5%. This suggests that using 0,46 as the threshold value for parameter t is an liberal choice that is likely to result in a number of false positives. For most trading pairs this would mean that the rug-pull determination would be only based on the identification of a rug-pull indicating transaction presented in algorithm 2.

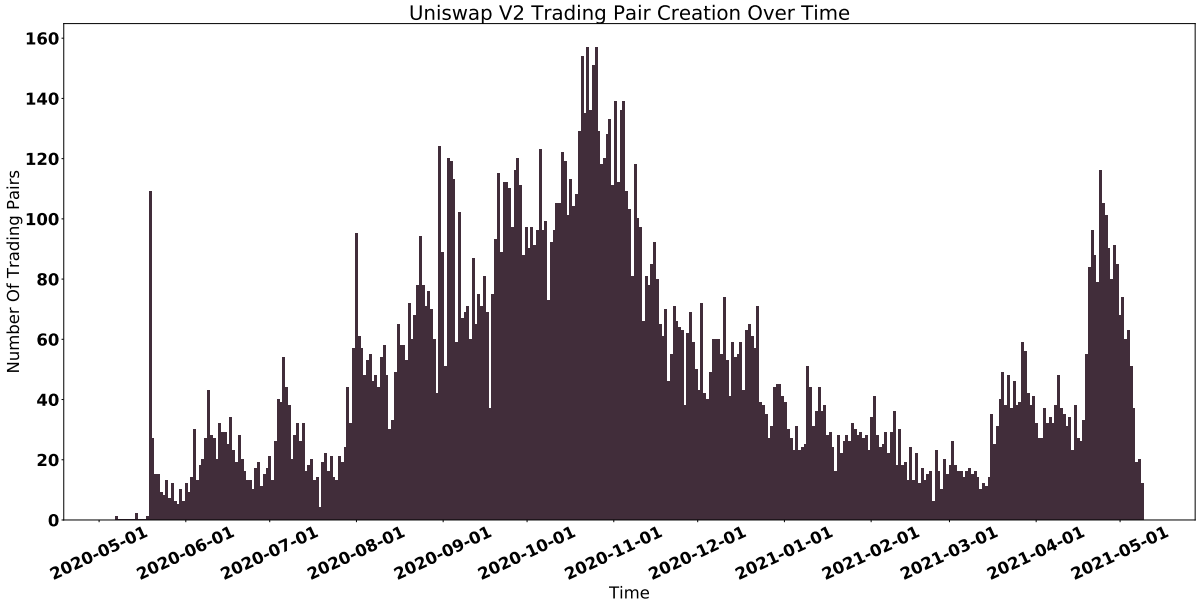

Figure 6.7: Uniswap V2 trading pair creations within the recorded time frame with one bar representing 24 hours.

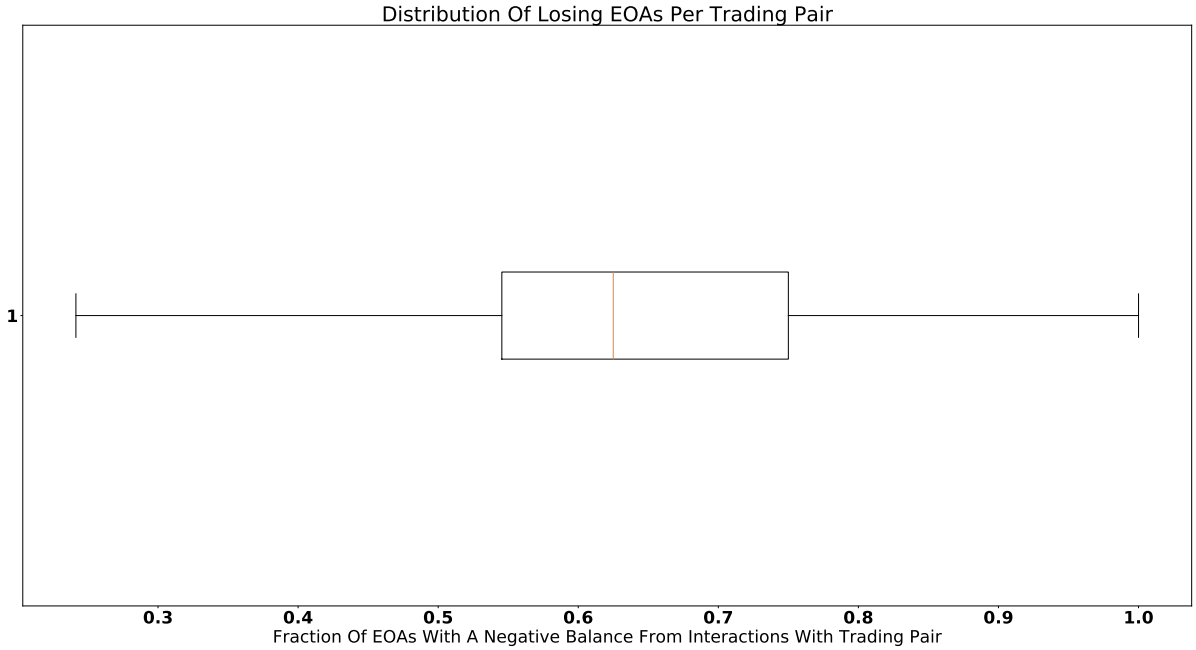

Figure 6.8: Distribution of fraction of EOAs with a negative base cryptoasset balance per trading pair.

6.2.2 Rug-Pull Estimate

Configured with optimal input parameter values our heuristic initially finds that 10.804 out of 18.421 eligible Uniswap V2 trading pairs are subject to at least one rug-pull within the recorded time frame. However, 13 of those pairs have only one interacting EOA which means that the pair creator is the only EOA to interact with the trading pair. For this reason we remove those trading pairs from the final result as it is illogical that the pair deployer executes a rug-pull on themselves. Thus in total **we find that 10.791 trading pairs are subject to a rug-pull which constitutes to 58,58% of the eligible trading pairs**. As illustrated in table 6.3, the significant differences between the mean and the median values of *sync events count*, *EOAs count* and *time until rug* are indicative of a long tail distribution. The majority of rug-pulls happen within 24 hours after the trading pair deployment on pairs with a low number of interacting EOAs and a small count of sync events. The relation between the baseline cryptoasset pool sizes before and after the rug-pull appears to be more extreme than the optimal threshold value derived from the parameter optimization process. Moreover, the glaring difference between mean and the median value of the *pool size relation* shows that the majority of WETH pools are depleted to a point where the remaining amount can be considered to be zero. What strikes as surprising is the fact that the mean value of the *fraction of losing EOAs* in pairs that are affected by a rug-pull is only approximately 4% higher than the fraction of losing EOAs in all eligible pairs presented in section 6.2.1.
In figure 6.9 we see the amount of rug-pulls per day in red and the amount of eligible trading pair creations per day in black. In the period between May 2020 and June 2020 the amount of rug-pulls is negligible because despite already being operational, Uniswap V2 is still largely unknown to the wider public at that time. However, starting from June 2020 we see an increase in both pair deployments and rug-pulls. The uptrend continues at an ever increasing pace until

it reaches its zenith at the end of October 2020. During this period that is partially referred to as *DeFi summer* the number of trading pairs deployments per day outperforms the amount of rug-pulls per day relative to the period after October. Before 1st November 2020, we have 9.881 trading pair creations and 4.885 rug-pull, thus a rug-pull to creation ratio of 0,494. After the 1st November 2020, we have 8.481 trading pair creations and 5.906 rug-pulls, thus a rug-pull to creation ratio of 0,694. In the second half of the recorded time frame, starting from November 2020, we see the absolute rug-pull numbers increase by 21%. Moreover, the rug-pull to pair creation ratio increases by 40%. This means that the probability to encounter a rug-pull in a recently deployed project becomes significantly higher after October 2020. Between January 2021 and March 2021 there are even days where Uniswap V2 faces more rug-pulls than trading pair deployments.

Name	*Mean value*	*Median value*	*Min. value*	*Max. value*
Sync events count	664	32	11	237.246
EOAs count	157	16	2	46.516
Fraction losing EOAs	68,0%	66,7%	46%	100%
Time until rug	404	15,9	0,0008	8449
Pool size relation	0,64%	4,2e-16%	6,7e-20%	8,6%

Table 6.3: Evaluation metrics of the rug-pull estimate. *Time until rug* - duration in hours between pair creation and rug-pull. *Pool size relation* - WETH pool size after rug-pull as a fraction of WETH pool size before the rug-pull.

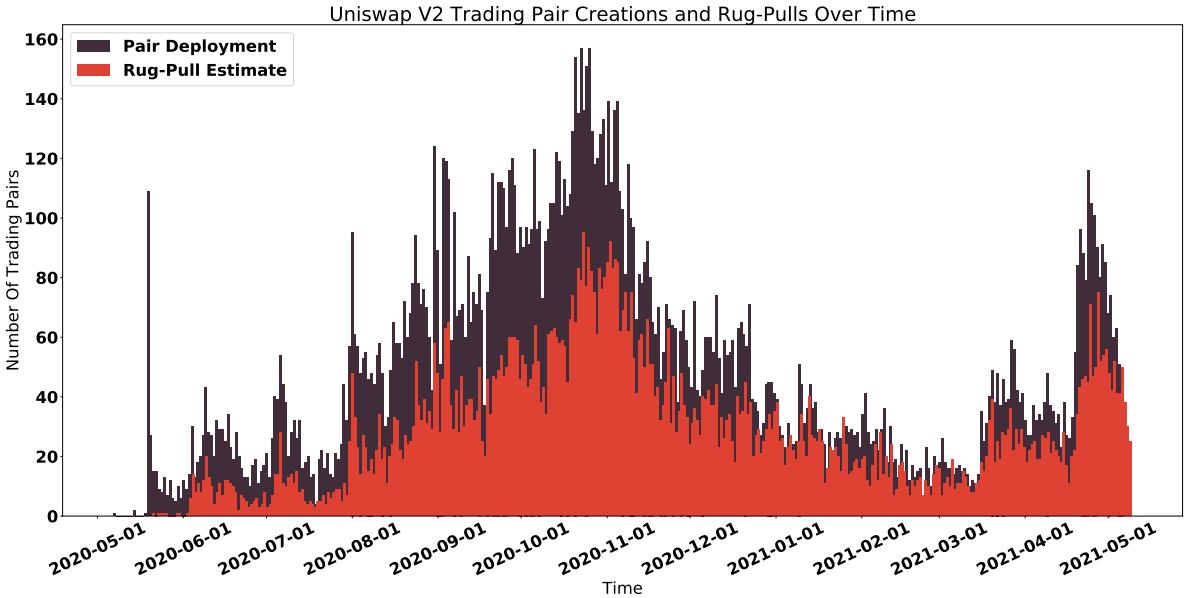

Figure 6.9: Uniswap V2 trading pair creations and estimated times of rug-pulls (overlaid plots) with one bar representing 24 hours.

Finally, we select the top 20 trading pairs measured by the number of sync events where a rug-pull has been identified. Using the sources presented in section 4.6, we validate our estimate by manually confirming the rug-pulls in those pairs. While a summary of the manual

validation is presented in table 6.4, more details are available in the attached data. We can confirm that 8 out of the 20 samples are indeed rug-pulls as there is sufficient proof in the form of media coverage and social media posts. We can attribute 4 samples to being so-called *pool migrations* which means that the current trading pair is abandoned and its liquidity transferred to a new trading pair. We provide a detailed explanation of this phenomena in section 7.1, but for now it should suffice to know that from a pool size based on-chain perspective a pool migration is indistinguishable from a rug-pull. For one sample, the *Earn Defi Coin*[1] we are not able to arrive at a conclusive result. The majority of communication about this project appears to happen in Chinese and it is unclear whether the project is simply performing exceptionally bad, or if it is the case of a pool migration. Nevertheless, there is a working website and an active social media presence. At last, we find that 7 rug-pull attributions are incorrect as the underlying cryptoassets are not only still active, but in some cases even reasonably successful judging by their market capitalization.

Manual examination results	*Number of samples*
rug-pull	8
pool migration	4
no rug-pull	7
inconclusive	1

Table 6.4: Manual examination of top 20 rug-pull trading pairs identified by the liberal estimate

[1] address: 0x2a79c9d77ba02b5e88c0f4057da378521809e11459f8b554092bfb9ada615bb6

7 Discussion

In this chapter we will discuss the results of our work as well as possible implications.

7.1 Insights from the Ground Truth Data Set

The ground truth data set we work with is unquestionably small in size. It is therefore doubtful whether a subset with a total of 80 samples can be considered as representative of a data-set with a total length of 35.610 samples. In fact, some input parameters which are optimal in the ground truth data-set, are exceedingly optimistic in the overall data set as explained in section 7.2. The relatively small size of the data-set also renders a split into a training set and a test set to be largely ineffective. Nevertheless, our ground-truth data set exhibits a very high degree of validity. Each sample is manually researched and confirmed as detailed in the attached data.
What strikes as especially interesting among the samples of our ground truth data set is the Uniswap V2 trading pair *SUSHI-WETH* which is part of the positive samples despite the fact that the cryptoasset *$SUSHI* is located within the top 100 cryptoassets measured by their market capitalization in October 2021. To find the reason for this apparent inconsistency, we need to explore the specifics of one of the most prominent contemporary rug-pulls as detailed in [76]. *$SUSHI* is the cryptoasset of the popular decentralized exchange *SushiSwap* which is created as a fork of Uniswap V2 in August 2020. By incentivising LPs the DEX is able to gain tremendous popularity within a very short time frame. The popularity of SushiSwap directly translates into a stern increase in the value of its cryptoasset *$SUSHI*. The sky-rocketing price leads its anonymous founder only known under the alias *chef Nomi* to drain the liquidity of the Uniswap V2 *SUSHI-WETH* pair not even two weeks after the trading pair is deployed. This rug-pull nets *chef Nomi* a profit of approximately 14 million US Dollars for a months work, but it also causes an outrage within the cryptoasset community as thousands of investors face catastrophic losses. Despite *chef Nomi's* best efforts to argue that the rug-pull was in fact a legitimate pay-out for their hard work, the captured funds are returned within a week and the ownership of SushiSwap is transferred away from *chef Nomi* as influential cryptoasset-funds solve the situation behind closed doors. In the aftermath of these events the original trading pair is abandoned and the liquidity is migrated to a new SushiSwap smart contract. *SushiSwap* itself continues to be one of the most successful DEX in existence and any investors that did not sell their *$SUSHI* during the turmoil are now in profit on their investment. While this story is interesting on itself, the part that is especially relevant for us is the pool migration. A pool migration is the event where a project abandons one CPMM DEX trading pair and deploys a new CPMM DEX trading pair or multiple trading pairs for the same cryptoasset. This also means that the project owners remove the liquidity from the old trading pair and transfer it to the new one. For our heuristic this act of pool migration is indistinguishable from a true rug-pull as the liquidity is pulled without ever recovering. As illustrated in table 6.4, four large trading pairs

that identified as rug-pulls during the estimate, are in fact pool migrations. We must therefore acknowledge that pool migrations as a source of errors for our heuristic and we will be looking to develop a solution for this problem in the future.

7.2 Insights from Parameter Optimization

Comparing the ground truth data set with the overall data set in regard to the specification of input parameters, we can see significant differences. The optimization process presented in section 6.1 delivers good results with a f1-score and an accuracy score of above 0,9. However, the ground truth sample may not be representative of the overall data set due to its small size. This becomes especially evident at the example of input parameter t, thus the threshold value for the fraction of EOAs with a negative baseline cryptoasset balance. Using the threshold values t = 0,46 and p = 0,06, we get a f1-score of 0,966 with 0 false positives. While there are multiple input parameter combinations that deliver them maximum f1-score, we choose the combination with the highest values in order to be as restrictive as possible without sacrificing f1-score quality. The high f-1 score suggests that the chosen parameter values are conservative in the sense that they do not allow for any wrongful rug-pull attributions. But looking at figure 7.1 it is clear that the supposedly optimal threshold value t is only optimal in the context of the ground truth data set. In the context of the whole data set a threshold value of t = 0,46 is not a conservative, but a very liberal parameter configuration as over 75% of all samples exhibit a fraction of losing EOAs of at least 46%. This means that the validation algorithm 3 provides a positive signal for most trading pairs. Since the identification of a rug-pull indicating transaction in algorithm 2 can result in wrongful attributions as explained in section 4.5, we can expect a number of false positives if we apply the threshold value t = 0,46 for the analysis of the wider data set. This effectively means that we provide a *high-bound* estimate of the number of rug-pulls that occur on Uniswap V2 within the recorded time frame.

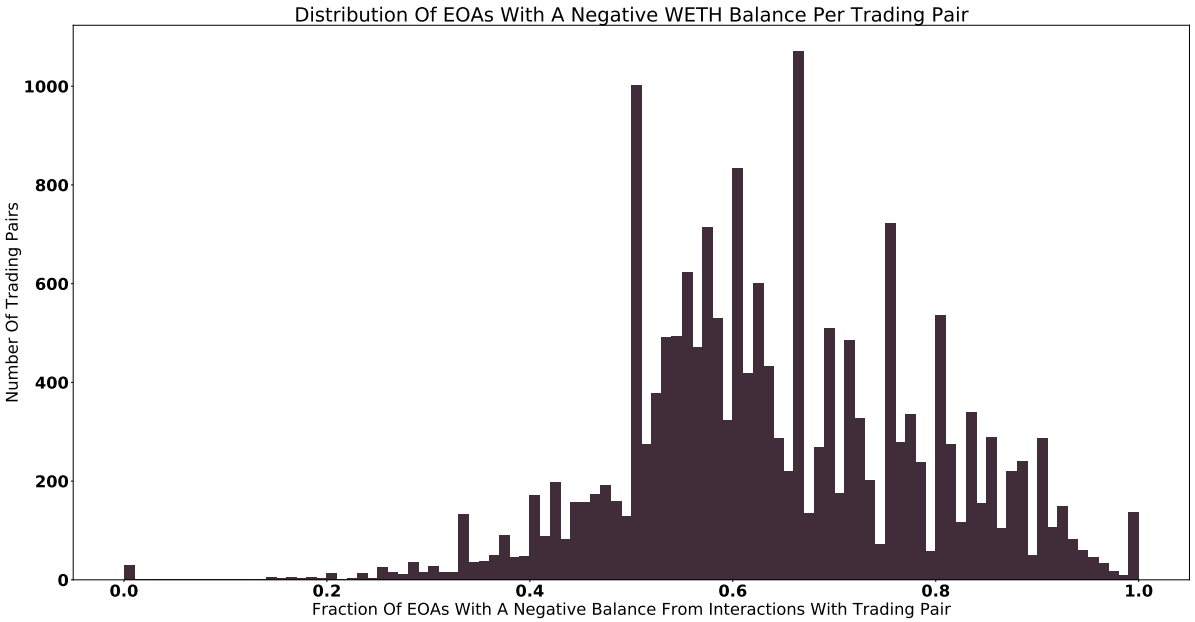

Figure 7.1: Distribution of fraction of EOAs with a negative base cryptoasset balance per trading pair.

7.3 Insights from Rug Pull Identification

Through the application of our rug-pull identification heuristic we find that in the period from May 2020 until May 2021 a significant fraction of trading pairs on Uniswap V2 are subject to rug-pulls. Specifically, we arrive at the high-bound estimate that approximately 58,58% of the listed pairs exhibit a catastrophic *WETH* liquidity implosion that does not recover within the recorded time frame. Moreover, a substantial number of EOAs exhibit a negative *WETH* surplus from their interactions with the identified trading pairs in the recorded time frame. The result lies within our expectations and resonates with the result of [69] that attributes approximately 50% of existing liquidity pools to be of fraudulent nature. Nevertheless, the set of samples upon which our results are based is quite different. Our recorded time frame is almost twice as long, but more important is the fact that [69] does not implement any sort of preprocessing. On the other hand, we disregard almost 50% of our samples as we demand that the processed trading pairs exhibit a minimum amount of sync events in order to ensure that these are true trading pairs and not just smart contracts deployed for testing or experimental purposes. It is however unlikely that 17.189 pairs with 10 or less sync events are listed on Uniswap V2 solely for testing or experimenting. To find reasons for the large number of such samples appears to be an interesting subject for future research. The general descriptive metrics we collect, such as the number of EOAs or the number of sync events per trading pair, all follow a heavy tailed distribution which is a typical distribution pattern for data sets obtained from the Ethereum ledger[48][54][68][69]. A small subset of trading pairs occupies the majority of available liquidity and the attention of investors. The only exception is the distribution of the fraction of EOAs with a negative baseline cryptoasset balance. Herein, we find the resemblance of a normal distribution with a slight shift as illustrated in figure 7.1. If the majority of EOAs represent individual investors, then it is logical that the distribution of their balances follows a symmetrical pattern since each investor can be presumed to be a generally independent variable.

Looking at the actual rug-pull identification, there are several noteworthy findings. First of all, we can confirm the results of [69] which state that fraudulent cryptoasset exhibit a short lifespan. According to our own results, over 50% of rug-pulls occur within 24 hours after trading pair deployment. This also confirms our high level reasoning presented section 4.2 that identifies speed of execution as an important characteristic of a rug-pull. However, there are outliers where the span between deployment and rug-pull can take up to 12 months. A manual examination of the top 20 of those pairs reveals that all of them exhibit low numbers of sync events with intervals of increasing length between them. While we cannot categorically rule out real trading activity, the number of affected EOAs and lost funds isn't noteworthy. It may appear as surprising that the fraction of losing EOAs in the affected pairs found by the estimate is only 4% higher than the global fraction of losing EOAs. But given the fact that the estimate finds a rug-pull in 58,58% of all samples, it is compelling for the averages to remain in close proximity to each other. It also suggests that the estimate contains a number of false positives. The metric of *pool size relation* presented in table 6.3 and table **??** confirms our initial expectation that the liquidity draining in a rug-pull is exceedingly severe.

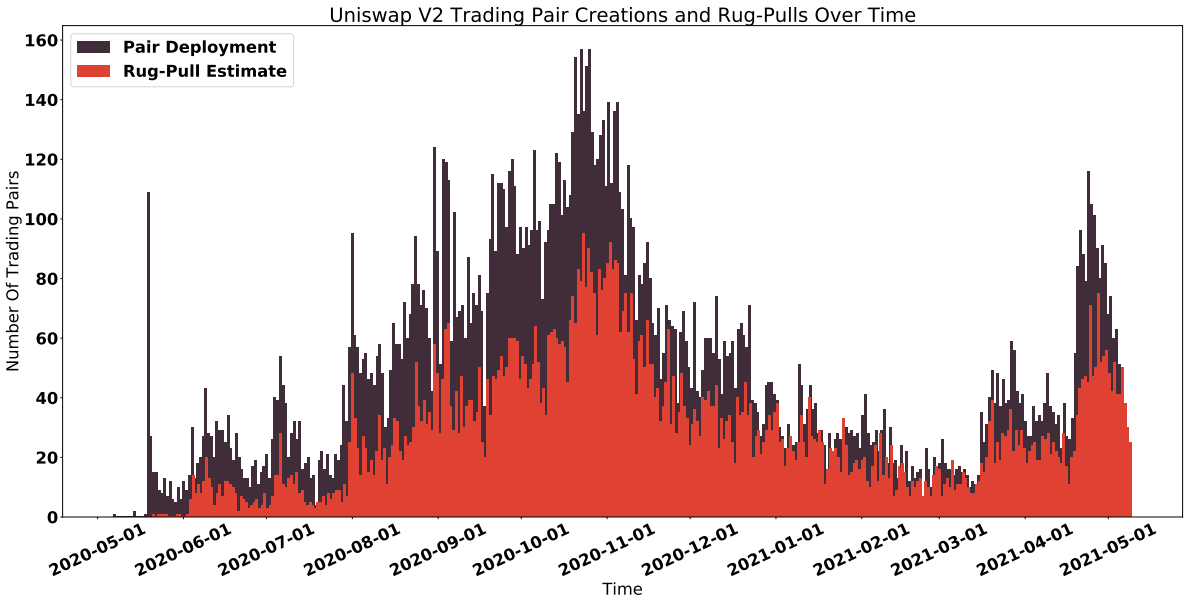

Figure 7.2: Uniswap V2 trading pair creations and estimated times of rug-pulls with one bar representing 24 hours.

Now let us consider the development of trading pair deployments and the rug-pull estimates over time. The first point of interest is the period of increased activity that lasts approximately from August until the end of October 2020. In 2019 [34] points out that the main use case of cryptoassets that run on the Ethereum ledger constitutes to buying and selling. At that time, the ultimate reason to hold a cryptoasset is to speculate that its value will increase in the future. In the summer of 2020 this narrative is challenged with the emergence of projects like *Compound Finance*[1] or *Yearn Finance*[2] that deploy on Uniswap V2. Without going into too much detail, the smart contracts developed by these projects enable their cryptoassets to become *productive assets*. For instance, depositing the cryptoasset $YFI in the *Vault* smart contract of Yearn Finance generates a yield over time. By keeping a certain amount of $YFI in the *Vault*, an investors can periodically withdraw the yield, which is paid out in $YFI, and sell it on the market. The ability to generate cash flows with cryptoassets is very well received and the valuations of these assets explode. For instance, between 20 July and 20 September 2020 the price for one $YFI increases from approximately 1.000 USD to 33.000 USD[3]. However, this alone does not explain the steep increase in trading pair deployments between August and November. On 17 September 2020 the Uniswap Project deploys its cryptoasset $UNI and distributes a total 150 million $UNI to EOAs that interacted with a Uniswap smart contract before 1st September 2020 which amounts to 400 $UNI per EOA[77]. At the time of the *airdrop*, the price for one $UNI is approximately 3,44 USD. Many thousands of Uniswap users suddenly find themselves in the possession of a substantial amount of funds. Moreover, public life is heavily restricted in many countries due to the ongoing COVID pandemic at that time. The promise of productive assets, the gifted funds and a lockdown induced boredom are the factors that are likely responsible for the Cambrian explosion of trading pairs on Uniswap V2 as illustrated in figure 7.2. While the amount

[1] https://compound.finance/
[2] https://yearn.finance/
[3] https://www.coingecko.com/en/coins/yearn-finance

of rug-pulls remains initially low, this development is causes an increase in rug-pulls as malicious actors can easily lure unsophisticated investors into their schemes especially since many investors are not really attached to the airdropped funds they never had to consciously work for. But as the number of transactions on Ethereum grows, so does the network congestion and thereby the gas price. And as the transactions require an ever increasing amount of $ETH, the demand for it rises and so does its price. As illustrated in figure 7.3, the price of Ether increases from 400 USD in November 2020 to 1200 USD in January 2021.

Figure 7.3: Development of $ETH price in United States Dollars between June 2020 and May 2021, sourced on 08.12.2021 from https://www.tradingview.com/.

Furthermore, we can see that the *Bitcoin dominance* increases from 60% in September to 70% in December 2020 as illustrated in figure 7.4. Bitcoin dominance represents Bitcoin's share in the total cryptoasset market capitalization and is a measure for the performance and popularity of *altcoins*, thus cryptoassets that are not Bitcoin. So starting from November 2020 the fees for transactions on Ethereum become an ever more significant expense while the cryptoasset market also shifts its focus towards Bitcoin that rallies from 20.000 USD to over 50.000 USD. This explains the serious drop in trading pair deployments illustrated in figure 7.2. It also explains why it is much more likely to encounter a rug-pull between November 2020 and February 2021 as detailed in section 6.2.2. Legitimate projects have little incentive to launch at a time where their cryptoasset can be expected to perform poorly. On the contrary, malicious actors have every incentive to execute as many rug-pulls as possible since the value of the captured funds keeps increasing, while the number of investors who can afford to use Uniswap V2 keeps decreasing. Finally, the increase in pair deployments towards the end of the recorded time frame can be explained by the huge drop of Bitcoin's dominance and the rotation of funds back to *altcoins*. However, a detailed examination of these dynamics goes beyond the scope of this work.

Figure 7.4: Development of Bitcoin's share in the total cryptoasset market capitalization between June 2020 and May 2021, sourced on 08.12.2021 from https://www.tradingview.com/.

Now that we have provided an understanding as for why the general development of trading pair deployments and rug-pulls happens the way it does, we can discuss the the manual examination of the most significant trading pairs where a rug-pull is attributed. As detailed in table 6.4, we can see that the liberal estimate exhibits a rather poor performance with 7 clearly wrong attributions. One of those wrongful attributions is the *BAND-WETH* trading pair. *Band Protocol* is an active and ongoing project[4] and its *BAND-WETH* trading pair on Uniswap V2 has remained in active use up until today which excludes the possibility of a pool migration. So the question persists as for how an active and even reasonably successful cryptoasset can be considered as a rug-pull. On 5 May 2020 Ether's price amounts to 205 USD and $BAND is worth approximately 1 USD, so the BAND-ETH exchange rate amounts to 0,0049. But as the narrative of *productive* cryptoassets unfolds, $BAND increases in price up to 16,5 USD on 1st September 2020, while $ETH only achieves a valuation of 490 USD which results in the BAND-ETH exchange rate of 0,0337. The tremendous impact of the *DeFi Summer* is easily visible in figure 7.5 and for the rest of the recorded time frame $BAND is not able to achieve that level of performance against Ether.

[4] https://bandprotocol.com/

Figure 7.5: Performance from USD value on 5 May 2020 to USD value on 5 May 2021 for $BAND (blue) and $ETH (orange), sourced on 08.12.2021 from https://www.tradingview.com/.

In section 2.4.1 we explain that AMMs rely on arbitrageurs to synchronize a trading pair's exchange rate to the global market. This means that the pool sizes of a trading pair are a reflection of the market wide dynamics. Thus an increased demand for $BAND relative to $ETH leads to an increase of the WETH pool size in the BAND-WETH trading pair. Furthermore, the Uniswap's significant increase in popularity over the summer of 2020 likely results in an overall pool size increase across all major trading pairs.

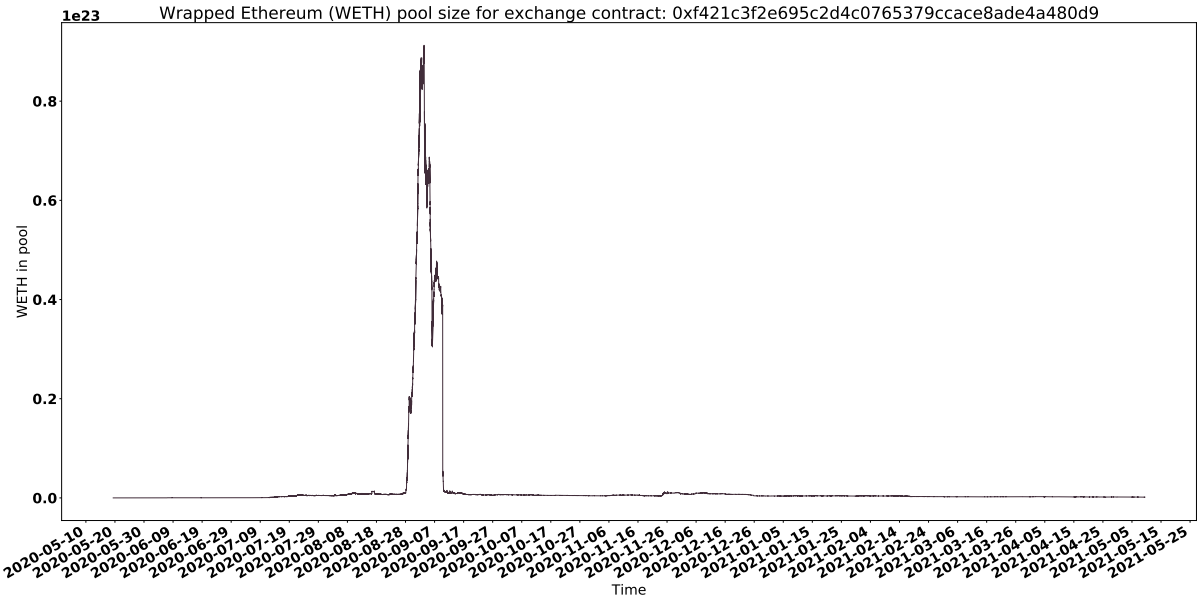

Figure 7.6: WETH pool size development in the Uniswap V2 BAND-WETH trading pair

But neither the increased popularity, nor the exceptional performance of $BAND can explain what we see in figure 7.6. Approximately on the 28 August 2020, the WETH pool size of the BAND-WETH pair increases from a few hundred Ether to over 9.000 Ether within a few days. 9.000 Ether amount to of over 3.000.000 USD at that time. Within approximately two weeks, the WETH pool size returns to its previous or usual levels. The retrace is not as stellar as the rise and happens in three stages. Our heuristic perceives these events as a rug-pull, since the average WETH pool size remains a fraction of what it was up until the end of the recorded time frame. However, we find no traces on the web of any sort of irregular activity within and after those two weeks. In USD terms, $BAND increases from 12,7 USD on 28 August to 15.5 USD on 2 September and retreats back to 9 USD on 7 September. Thus while significant volatility is present, this volatility is not extraordinary for that asset. Ultimately, we see two possible explanations for this pool size spike. It is theoretically possible that a high net worth investor decides to buy over 3 million USD worth of $BAND on Uniswap. But this scenario is very unlikely, because the exchange rate for these swaps would be beyond abysmal as the trading pair simply lacks the liquidity to facilitate an exchange of such volume. A much more likely scenario is a high net worth investor that decides to provide liquidity for the BAND-WETH trading pair, because being an LP on popular trading pairs can be very lucrative. However, both $BAND and $ETH begin to retrace after 1st September as illustrated in figure 7.5. This may cause the investor to remove the provided liquidity which brings the pool size back to its usual levels. In the course of the manual examination presented in table 6.4, we find similar pool size dynamics in all pools where a rug-pull is wrongfully attributed except for cases of pool migrations. While the examination of large volume liquidity provisions and their effects appears as an interesting topic for future research, we must acknowledge that our heuristic may mistake these pool size structures for rug-pulls.
In conclusion, our rug-pull identification heuristic provides an overall good outline over the dynamics on the Uniswap V2 DEX between May 2020 and May 2021. Both the general metrics and the rug-pull specific metrics that we derive fall in line with the results of previous research about Ethereum and are also in sync with the general cryptoasset market dynamics during that time.

7.4 Limitations

While we are able to provide a reasonably specific picture about rug-pulls on Uniswap, we face a series of limiting factors. First of all, we need to acknowledge that the basis for our input parameter optimization is thin. For a future iteration of this work, it would be desirable to significantly extend the existing ground truth data set. Next we need to address the *resource conscious implementation* presented in section 4.7.2. We utilize this resampling procedure for algorithm 2 both during parameter optimization and the subsequent analysis. Herein, the reduction of sync events in affected trading pairs ranges from approximately 50% in smaller pairs to 99,9% in the largest pair. Nevertheless, we are still able achieve good results during parameter optimization and our overall results are consistent with previous research. However, we do not know the extent of the effect that the resampling procedure has. At last, we can discuss the fundamental limitations of our algorithms. What we essentially measure is the strength of the contraction of the baseline cryptoasset liquidity in a CPMM DEX trading pair. This feature is essential to a rug-pull event, but it is not exclusive to it. As we discussed earlier in this chapter, there are certain circumstances where an extreme reduction in liquidity can occur without

having a detrimental effect on the investors. In the case of pool migrations, we are unable to differentiate them from rug-pulls as these operations are communicated off-chain through social media. Of course, we could monitor whether the deployer of the old trading pair creates a new trading pair in the temporal vicinity of the alleged rug-pull. But pool migrations can happen across DEXs and even across chains, thus making this approach to so complex that it becomes infeasible. The other pool size related error source is the excessive, but temporary increase in the baseline cryptoasset pool size. This can occur both due to swaps and due to liquidity provision. The former case is presumably rare, as executing a swap that is many magnitudes larger than the available liquidity is a very bad deal for the initiator. However, the latter case is entirely reasonable and is thus a contributing factor to the number of false positives in our results.

7.5 Types of Rug-Pulls

Our heuristic queries on-chain data to identify the outcomes of rug-pulls in CPMM DEX trading pairs. However, as mentioned in section 4.5, there are many different ways to facilitate a rug-pull. In this section we present a selection rug-pull types ranging from the original *modus operandi* of a rug-pull to rug-pulls instigated by external events. The first two types are also corroborated by the results of [69].

7.5.1 Liquidity Removal

The first type of rug-pull is the very reason why this term exists. As described in chapter 1, for this type we need an entity to be in possession of a significant fraction of the available liquidity in a trading pair. Typically, this entity is the deployer of the trading pair, but it's not a necessary requirement. Nevertheless, the impact of this type of rug-pull is much greater when it is executed by the contract deployer as it destroys the confidence in the future development of the affected project. Furthermore, the entity must have unrestricted access to their LP tokens. The actual rug-pull consists of two parts. First the entity burns its LP tokens thereby removing its stake from the pools of the trading pair. Then it swaps some or all of its returned cryptoassets for the remaining *base* cryptoasset liquidity. Other trading pair participants are now faced with an nigh empty base cryptoasset pool, thus rendering their investments close to worthless. However, the implosion of liquidity does not automatically need to be final. As described in the previous section, the example of SushiSwap shows that even after a liquidity removal by the project owner, a cryptoasset can make a comeback given the right backing.

7.5.2 Smart Contract Vulnerability

[64],[63],[69], [68] and [61] find that functions within smart contracts of cryptoassets can be harmful to the entities interacting with malicious smart contracts. This opens doors to a huge variety of attack vectors on the liquidity in a AMM trading pair. For instance, the assets *HOHO* and *HATCH* from section 4.1 both have a function that allows the smart contract owner to create an arbitrary number of tokens. The perpetrators then swap the tokens they create out of thin air for the base cryptoasset liquidity in the trading pair. A more recent example of a rug-pull implemented through smart contract vulnerabilities is the rug-pull on a project called *SQUID*

which alleged to be connected to the popular television series *Squid Game*[78]. Herein, the vulnerability constitutes to the fact that the smart contract allows only a few selected accounts to swap the *SQUID* cryptoasset for the base cryptoasset liquidity. Thus an unsuspecting investors is able to buy the *SQUID* asset, but can never sell. In the beginning, the design of this scheme results in a sky-rocketing price appreciation, but unfortunately for all other investors the perpetrators who have access to the white-listed accounts eventually do drain the base cryptoasset liquidity. Unlike the previous type, a liquidity implosion caused by smart contract vulnerabilities is usually final. When an entity knowingly deploys a cryptoasset with vulnerabilities, the reasons for the existence of said asset can only be of malicious nature. This means that the underlying smart contract is very unlikely to offer any real innovation or value as a product or as a service turning any efforts to revive the project after the rug-pull to be fruitless.

7.5.3 External causes

Regarding the previous types, individuals with sophisticated technological knowledge may be able to assess the risk of a rug-pull by examining the underlying smart contract and LP tokens. However, the risk of a rug-pull caused by external causes cannot be assessed with on-chain data. For instance, on 6 February 2021 the cryptoasset *$DMG* which belongs to the project *DeFi Money Markets* loses 90% of its value within 1 hour and does not recover the remainder of the recorded time frame. The reason for this lies in the fact that the legal entity of the project is charged by the United States Securities and Exchange Commission with sale of unregistered securities[79]. Consequently, the owners of the project cease any and all operations immediately resulting in catastrophic losses for its investors. From an on-chain perspective this sequence of events does not differ from deliberately caused rug-pull types described above.

7.6 How to deal with rug-pulls

The results of our work show that rug-pulls are a risk for users of AMM DEXs. The fact that we identify rug-pulls in over 50% of the trading pairs listed on Uniswap V2 clearly shows that this is a significant problem that should not be ignored when conducting business on AMMs. In this section we first propose a series of best practices for individual investors and later discuss how rug-pulls can be prevented from an institutional perspective. It should be noted that the measures we propose are focused on deliberate rug-pulls caused by malicious intent. Rug-pulls that originate from external causes are not in the scope of this discussion.

7.6.1 Perspective of individual investors

As [61], [63] and [69] point out, the best protection for individual investors is profound technological understanding of the subject at hand. Ideally, an investor that operates on AMMs should be able to read and understand the source code of smart contracts. Moreover, they should have a good understanding of the distributed ledger technology in general in order to explain the implications of smart contract functions within the network. As section 4.1 shows, rug-pulls are generally not exceedingly sophisticated and individuals with relevant technological knowledge and enough time should be able to spot malicious crypto assets easily. Unfortunately, these prerequisites are not attainable for the majority of investors as it can take years to

develop the needed skills for a person that has no background in computer science or a similar discipline. Therefore, we will now propose a series of best practices that can be implemented by any investor and require no technological background.

7.6.1.1 Market capitalization as a measure of trust

As a rule of thumb one can remember that the risk of a rug-pull and the market capitalisation of a cryptoasset resemble an inverse correlation. Thus investing in a cryptoasset with a market capitalisation of 100.000 USD is much riskier than investing in a cryptoasset with a market capitalisation of 1.000.000.000 USD. This rule of thumb generally holds true for any market, but is especially important in the cryptoasset market. As the example of *Sushi Swap* shows, rug-pulls on popular cryptoassets can result in serious efforts to apprehend the perpetrators. Moreover, the public nature of smart contracts makes it possible for anyone to examine the contract. Thus the likelihood of flaws being exposed grows larger as the cryptoasset becomes more popular. However, rug-pulls on insignificant cryptoassets that have a few hundred go largely unnoticed. Therefore there is an incentive for malicious actors to pull the rug before their scheme becomes too big. For individual investors this simply means that investing in *micro-caps*, thus cryptoassets with a very small market capitalisation, is an extremely risky endeavour. When investing in a micro-cap on an AMM DEX, the risk definition should always put an emphasis on the possibility of a total loss of funds due to a rug-pull. Determining the exact relation between the market capitalisation of a cryptoasset and the risk of a rug-pull can be a promising topic for future research.

7.6.1.2 Project's founders as a measure of trust

The example of the *RIFT* rug-pull from section 4.1.5 shows that malicious actors can use the role of the founder to create trust and instil false hopes into their scheme. Thus researching the founders is a viable, non-technical vetting strategy for new projects. Herein, the first step is to find out whether the founding team chooses to remain anonymous. Naturally, a cryptoasset that is being developed by an anonymous team is a much riskier investment than a cryptoasset where the details about the developers are publicly available. Nevertheless, this does not mean that all anonymous developers act in bad faith. For instance, many core developers of Monero *$XRM* remain anonymous even after contributing to the project for many years[80]. Unfortunately, it is not enough to simply confirm that personal information on the developers of a project is available. It is also necessary to verify this information as malicious actors can create fake identities in order to attract investors who refuse to engage with anonymous projects.

7.6.1.3 Do your own research

The existence of over 30.000 trading pairs on Uniswap V2 within a year after its creation is a good indication of the speed at which the cryptoasset market moves. This explosive growth coupled with a high degree of decentralization leads to the fact that thorough, independent assessment of projects often does not exist. This is especially true for new projects with a low number of investors. On the other hand, projects must grow quickly in terms of users and market capitalization in order to not fade into irrelevance. The need to capture value from investors

who often do not fully understand the assets they are investing in and the fast pace of social media lead to an unhealthy dynamic. Projects are heavily incentivised to over-promise and under-deliver as the public perception of a cryptoasset is often more important for its valuation than the actual service or product it delivers. Unlike companies that are listed on regulated exchanges, cryptoasset projects barely face any restrictions or regulations in regard to their reporting. This means that information made available by the projects themselves should never be trusted blindly. Instead, cross-verification via social media research or source code assessment is needed. Another viable approach to verify the claims of projects is the study of audit reports. Smart contract auditing services is a growing branch of the cryptoasset industry[81]. It is a promising sign when a project chooses have the source code of its cryptoasset audited by an independent service, but it is even better when the audit is done by multiple services in parallel. These reports can provide non-technical investors with valuable information about possible smart contract vulnerabilities. Unfortunately, the auditing services themselves are also forced to deliver results very quickly as they are pressured by their clients. This results in audited smart contracts being faced unforeseen errors or exploits on a regular basis[82]. Nevertheless, a malicious cryptoasset that is created solely for the purpose of a rug-pull is highly unlikely to subject itself to an audit. Finally, the third major source of information for individual investors are opinions about cryptoassets found on the web. This includes news articles, assessments published by funds or similar organizations, supposedly educational content published on social media or websites and opinions of social media users. This source suffers from the same problems as information published by projects themselves. The actors involved are not bound by regulations to disclose their level of involvement with the cryptoasset they are publishing about. It is thus impossible to know their true motivation for publishing their opinions. A fund might share extremely positive assessments of a cryptoasset only to generate their exit liquidity[5]. A fund might share extremely negative assessments of a cryptoasset with the intention to create an atmosphere of fear, uncertainty and doubt only to buy the asset at a lower price[83]. Individual social media influencers are regularly paid by projects to publish positive assessments about them[84]. Naturally, most influencers do not disclose that their opinions are actually paid promotions. This is especially relevant for malicious projects as paying influencers is a relatively cheap and yet effective strategy for generating hype to lure in investors. As illustrated in section 4.1, most rug-pulls are initially promoted by social media influencers as safe and lucrative investments. All in all it is of utter importance to do independent research on potential investments. Being distrustful is a virtue in the market of cryptoassets. A sophisticated investors will not only assess the quality of information about a cryptoasset, but also the motivation of the ones publishing said information. Any and all alleged facts or opinions should be cross-checked and verified in order to arrive at a truly independent opinion about a potential investment.

7.6.2 Perspective of institutions

Before going into the details of the institutional perspective, we need to clarify which entities we consider to be *institutions* in the context of decentralized finance. An institution is a mutually accepted system of practices that creates and alters the rules and norms in a

[5] In order to sell a large position in a cryptoasset without crashing its price and thus suffering from slippage, there needs to be an increased numbers of buyers while the position is exited.

community[85]. In the context of DeFi, there are two such accepted systems of practices. First, there are the distributed ledgers themselves. Ethereum is possibly the most important institution in this regard as it not only sets the rules for operation within its own boundaries, but also serves as an example to other distributed ledgers due to its first-mover status. The second type of institutions are decentralized exchanges as they facilitate the currently most important use case for cryptoassets which is speculation[34]. Consequently, we first explore which measures can be included into distributed ledgers and then look into the measures that can be applied by individual DEXs.

7.6.2.1 Distributed ledgers

When trying to tackle the issue of rug-pulls on the ledger layer, we are faced with a fundamental dilemma. One of the defining features of distributed ledgers is the fact that they are *permissionless* and *decentralized*. In the case of Ethereum this means that anyone with access to the network can deploy any smart contract they like without facing regulation or oversight. Implementing measures against rug-pulls in the design of the ledger itself quickly leads to a limitation of the decentralized and permissionless state. For instance, if we force smart contract deployers to submit documents verifying their identity, then we would first of all limit the decentralization as these documents need to be stored somewhere. A possible way to circumvent this is to implement a zero knowledge proof infrastructure, but such systems do not exist yet[86]. Secondly we would effectively end the permissionless state by excluding all those without proper identification. Another approach to limiting rug-pulls is to regulate certain smart contract mechanics. Specifically, we could regulate the process of *minting* cryptoassets as this functionality is often used to facilitate rug-pulls. However, these regulations would likely need to be updated routinely to keep up with the progress of the ledger itself and with the efforts of malicious actors to circumvent them. This leads us back to the problem of centralized authority. A distributed ledger cannot be called permissionless, if there is a centralized entity that decides which smart contracts are allowed and which are not. The third possible approach to rug-pulls is not a preventive measure, but a reactive one as it entails a *hard-fork* after a rug-pull or a similar attack. In this context a hard-fork means the roll-back of the blockchain to a moment in time before the malicious event. [87] describes how such a roll-back is implemented in the aftermath of one of the largest exploits on Ethereum, the *DAO attack* in 2016. At that time, approximately 14% of all circulating *ETH* are in the hands of a malicious actor as the result of a smart contract vulnerability exploit. Faced with an existential threat, the *Ethereum Foundation* proposes a hard-fork in order to revert the damage. The issue with hard-forks on a distributed ledger is that the majority of node operators needs implement the respective software upgrade in order for the hard-fork to come in effect. But despite the critical nature of the situation, the proposed roll-back is perceived as highly controversial by the community and some node operators refuse to participate. Instead, they continue to run their nodes on the unaltered blockchain leading to the creation of the distributed ledger known as *Ethereum Classic*. The above example shows that rolling back the network whenever a rug-pull occurs is not a viable solution and it is unlikely that Ethereum will ever implement such a step again. To sum it up, we find that measures on the distributed ledger layer specifically designed to counter-act rug-pulls should not be implemented, if the ledger is to be called permissionless. Naturally, the next step would be a discussion about whether a permissioned distributed ledger is the superior solution altogether, but such a discussion goes beyond the scope of this work.

7.6.2.2 Decentralized Exchanges

The aspect of being permissionless is not as critical in regard to decentralized exchanges as a DEX only represents a subset of a decentralized ledger. Yet it remains questionable whether a permissioned DEX is a feasible business model in a market that is ideologically antipathetic towards centralization and control. Nevertheless we present a number of design features against rug-pulls that a DEX can incorporate into its smart contract system. These features can be implemented as a strongly permissioned approach where a trading pair is only listed if all the requirements are fulfilled. On the other hand, a weakly permissioned approach would assign a *trust score* to trading pairs based on the number of the following rug-pull preventing measures that the deployer chooses to implement:

- **Liquidity lock**
  As illustrated in section 4.1.4, liquidity locking services are already present on the Ethereum ledger. Generally, these are smart contracts that receive LP tokens and hold them under a time-lock until the agreed vesting period expires. In their current form, these are independent services that receive the LP tokens from the contract deployers themselves. A DEX could take this approach a step further by introducing an option to time lock the initially supplied liquidity at the moment of the trading pair deployment. This would prevent a significant number of unsophisticated rug-pulls where the deployers removes the initially supplied liquidity shortly after the launch of the trading pair. But as the example of *Hatch DAO* shows, a trading pair with locked liquidity remains vulnerable to rug-pulls facilitated with smart contract deficiencies.
- **Auditing**
  A DEX could partner with or acquire one or several smart contract auditing services and only list trading pairs of cryptoassets that successfully pass the audit. Unfortunately, this comes with a number of complications. If this mechanic is implemented in a smart contract as part of a strongly permissioned approach, but the partnership fails for some reason, then the DEX will no longer be able to list new assets. Moreover, the costs for listing new trading pairs would rise significantly, since smart contract audits are expensive and time consuming affairs[88]. Finally, the pressure to list new trading pairs continuously could impair the quality of the audits.
- **Deployer KYC**
  This measure would implement an identity verification process for trading pairs deployers. As thorough remote identity verification on an international level creates creates a huge overhead, this feature is likely to be outsourced to existing online identity verification services.
- **Insurance**
  Insurance protocols are a growing sector within the decentralized finance domain[89]. However, services like *Nexus Mutual*[6] only offer insurance for individual accounts. Specifically, EOAs that hold the *$NXS* cryptoasset are entitled to a share of the insurance pool proportional to the amount they are holding. A DEX could partner with such a protocol to offer insurance for trading pairs that implement rug-pull preventing measures. The respective costs could be offset through an additional swap fee.

[6] https://nexusmutual.io/

- **Security Deposit**
  Closely related to the previous measure, trading pair deployers could be asked to provide a security deposit that is held in escrow for a certain vesting period. The amount of the deposit would dependent on the initially supplied liquidity and short-term growth projections. If the trading pair is subject to a rug-pull shortly after its deployment, then the security deposit will be distributed among the affected investors.

8 Conclusion

In this work we introduce a deterministic, on-chain heuristic for the identification of trading pairs on the Uniswap V2 DEX that are subject to rug-pulls. Based on a series of introductory examples, we explore a high-level view of the rug-pull phenomena both from an on-chain perspective and from the point of view of affected investors. Having provided a general understanding of the term, we propose an indirect definition for the rug-pull event by characterizing the effects of a rug-pull on a CPMM DEX trading pair. Thereafter, we present a series of threshold based based algorithms that translate our rug-pull definition into a rug-pull identification heuristic. In order to determine the optimal threshold values, we create a ground truth data set consisting of Uniswap V2 trading pairs that are subject to a rug-pull. Using the ground truth data set, we find the optimal threshold values for the identification of rug-pulls on Uniswap V2. Equipped with the optimized input-parameters, we analyse all existing trading pairs in the period between May 2020 and May 2021. Our results show that rug-pulls constitute to a significant problem that should not be ignored by anyone who intends to use Uniswap V2. Finally, we propose a series of measures that individual investors can implement in order minimize the probability of being affected by a rug-pull. Moreover, we also recommend a series of rug-pull preventing measures that can be implemented by a CPMM DEX. As for future research, what strikes as promising is the application of our rug-pull identification heuristic beyond Uniswap V2. As the fundamental principle of asset pools bound by a pricing curve remains true for all CPMM DEXs, future research can use our heuristic to measure the impact of rug-pulls across DEXs and even across chains. Such application of our work can increase the size of our ground truth data set up to the point where the application of machine learning becomes feasible. Subsequently, features can be discovered that allow for the on-chain *prediction* of rug-pulls. This would provide great value both for academia and cryptoasset market participants.

List of Tables

List of Figures

Bibliography

[1] D. Birch, *Before Babylon, Beyond Bitcoin: From Money That We Understand to Money That Understands Us*. London publishing partnership, 2017.

[2] G. Hanhui and M. Jie, "The silver standard, warfare and inflation: The mechanism of paper money in yuan china," 2016.

[3] M. W. Paas and J. R. Paas, *The Kipper und Wipper Inflation, 1619-23: An Economic History with Contemporary German Broadsheets*. Yale University Press, 2012.

[4] P. Hallwood, R. MacDonald, and I. W. Marsh, "An assessment of the causes of the abandonment of the gold standard by the us in 1933," *Southern Economic Journal*, pp. 448–459, 2000.

[5] C. J. Zoeller, "Closing the gold window: The end of bretton woods as a contingency plan," *Politics & Society*, vol. 47, no. 1, pp. 3–22, 2019.

[6] A. Krishnamurthy and A. Vissing-Jorgensen, "The effects of quantitative easing on interest rates: channels and implications for policy," National Bureau of Economic Research, Tech. Rep., 2011.

[7] J. Hartley and A. Rebucci, "An event study of covid-19 central bank quantitative easing in advanced and emerging economies," *NBER Working Paper*, no. w27339, 2020.

[8] U. B. of Labor Statistics, "Cpi inflation calculator," 2021. [Online]. Available: https://www.bls.gov/data/inflation_calculator.htm

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," Manubot, Tech. Rep., 2009.

[10] A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.

[11] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[12] G. Wood and V. Buterin, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum project yellow paper*, vol. 151, no. 2014, pp. 1–32, 2014.

[13] P. B. et al., "An analysis of atomic swaps on and between ethereum blockchains using smart contracts," 2018.

[14] K. N. Pourpouneh, Mohsen and O. Ross, "Automated market makers," 2020.

[15] V. Mohan, "Automated market makers and decentralized exchanges: a defi primer," 10 2020.

[16] N. Z. Adams, Hayden and D. Robinson, "Uniswap v2 core whitepaper," 2020. [Online]. Available: https://uniswap.org/whitepaper

[17] Uniswap. (2020) Pool liquidity. [Online]. Available: https://uniswap.org/docs/v1/frontend-integration/pool-liquidity/#add-liquidity

[18] T. Khaitan. (2020, 9) Chef nomi pulls the sushi rug on sushiswap lps. [Online]. Available: https://defirate.com/sushiswap-rug-pull/

[19] N. Chong. (2020, 10) Trust no one: Founders of once-50 million usd ethereum coin "rug pull" users. [Online]. Available: https://cryptoslate.com/trust-no-one-founders-of-once-50-million-ethereum-coin-rug-pull-users/

[20] J. Willet, "Mastercoin complete specification," Tech. Rep., 2013. [Online]. Available: https://cryptorating.eu/whitepapers/Omni/MasterCoin%20Specification%201.1.pdf

[21] V. Buterin, "A prehistory of the ethereum protocol," *Vitalik Buterin's website*, 2017.

[22] A. M. Antonopoulos and G. Wood, *Mastering ethereum: building smart contracts and dapps*. O'reilly Media, 2018.

[23] V. Buterin, "A next generation smart contract & decentralized application platform (2013) whitepaper," *Ethereum Foundation*, 2013.

[24] C. Dannen, *Introducing Ethereum and solidity*. Springer, 2017, vol. 318.

[25] V. Gupta, "The ethereum launch process," Ethereum Foundation, Tech. Rep., 2015. [Online]. Available: https://blog.ethereum.org/2015/03/03/ethereum-launch-process/

[26] V. Buterin, "Ethereum research update - metropolis," Ethereum Foundation, Tech. Rep., 2016. [Online]. Available: https://blog.ethereum.org/2016/12/04/ethereum-research-update/

[27] S. Tikhomirov, "Ethereum: state of knowledge and research perspectives," in *International Symposium on Foundations and Practice of Security*. Springer, 2017, pp. 206–221.

[28] V. Dhillon, D. Metcalf, and M. Hooper, "Unpacking ethereum," in *Blockchain Enabled Applications*. Springer, 2021, pp. 37–72.

[29] L. Burkholder, "The halting problem," *ACM SIGACT News*, vol. 18, no. 3, pp. 48–60, 1987.

[30] M. Zuidhoorn, "Why do we need transaction data?" 02 2019. [Online]. Available: https://medium.com/mycrypto/why-do-we-need-transaction-data-39c922930e92

[31] I. Asmundson and C. Oner, "Back to basics: What is money?: Without it, modern economies could not function," *Finance & Development*, vol. 49, no. 003, 2012.

[32] F. A. T. FORCE, "Guidance for a risk-based approach: virtual assets and virtual asset service providers. 2019," *www.fatf-gafi. org/publications/fatfrecommendations/documents/Guidance-RBA-virtual-assets. html. Acesso em*, vol. 9, 2019.

[33] R. H. Thaler, "Anomalies: Saving, fungibility, and mental accounts," *Journal of economic perspectives*, vol. 4, no. 1, pp. 193–205, 1990.

[34] F. Victor and B. Lüders, *Measuring Ethereum-Based ERC20 Token Networks*, 09 2019, pp. 113–129.

[35] G. Hileman and M. Rauchs, "Global cryptocurrency benchmarking study," *Cambridge Centre for Alternative Finance*, vol. 33, pp. 33–113, 2017.

[36] I. Makarov and A. Schoar, "Trading and arbitrage in cryptocurrency markets," *Journal of Financial Economics*, vol. 135, no. 2, pp. 293–319, 2020.

[37] R. Hanson, "Combinatorial information market design," *Information Systems Frontiers*, vol. 5, no. 1, pp. 107–119, 2003.

[38] C. Comerton-Forde, T. Hendershott, C. M. Jones, P. C. Moulton, and M. S. Seasholes, "Time variation in liquidity: The role of market-maker inventories and revenues," *The journal of finance*,

vol. 65, no. 1, pp. 295–331, 2010.

[39] F. Martinelli and N. Mushegian, "A non-custodial portfolio manager, liquidity provider, and price sensor," *URl: https://balancer. finance/whitepaper*, 2019.

[40] A. Capponi and R. Jia, "The adoption of blockchain-based decentralized exchanges," *arXiv preprint arXiv:2103.08842*, 2021.

[41] Y. Chen and C. Bellavitis, "Blockchain disruption and decentralized finance: The rise of decentralized business models," *Journal of Business Venturing Insights*, no. 13, p. e00151, 2020.

[42] G. Angeris and T. Chitra, "Improved price oracles: Constant function market makers," in *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, 2020, pp. 80–91.

[43] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, and D. Robinson, "Uniswap v3 core," 2021.

[44] Uniswap, "Uniswap documentation," 2021. [Online]. Available: https://uniswap.org/docs/v2

[45] R. Bulat, "Uniswap v2: Everything new with the decentralised exchange." [Online]. Available: https://rossbulat.medium.com/uniswap-v2-everything-new-with-the-decentralised-exchange-52b4bb2093ab

[46] J. Bacidore, R. H. Battalio, and R. H. Jennings, "Order submission strategies, liquidity supply, and trading in pennies on the new york stock exchange," *Journal of Financial Markets*, vol. 6, no. 3, pp. 337–362, 2003.

[47] J. E. Young, "On equivalence of automated market maker and limit order book systems," 2020.

[48] S. Somin, G. Gordon, and Y. Altshuler, "Network analysis of erc20 tokens trading on ethereum blockchain," in *International Conference on Complex Systems*. Springer, 2018, pp. 439–450.

[49] D. Guo, J. Dong, and K. Wang, "Graph structure and statistical properties of ethereum transaction relationships," *Information Sciences*, vol. 492, pp. 58–71, 2019.

[50] T. Chen, Z. Li, Y. Zhu, J. Chen, X. Luo, J. C.-S. Lui, X. Lin, and X. Zhang, "Understanding ethereum via graph analysis," *ACM Transactions on Internet Technology (TOIT)*, vol. 20, no. 2, pp. 1–32, 2020.

[51] X. T. Lee, A. Khan, S. Sen Gupta, Y. H. Ong, and X. Liu, "Measurements, analyses, and insights on the entire ethereum blockchain network," in *Proceedings of The Web Conference 2020*, 2020, pp. 155–166.

[52] L. Zhao, S. S. Gupta, A. Khan, and R. Luo, "Temporal analysis of the entire ethereum blockchain network," in *Proceedings of the Web Conference 2021*, April 2021.

[53] Y. Li, U. Islambekov, C. Akcora, E. Smirnova, Y. R. Gel, and M. Kantarcioglu, "Dissecting ethereum blockchain analytics: What we learn from topology and geometry of the ethereum graph?" in *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020, pp. 523–531.

[54] W. Chen, T. Zhang, Z. Chen, Z. Zheng, and Y. Lu, "Traveling the token world: A graph analysis of ethereum erc20 token ecosystem," in *Proceedings of The Web Conference 2020*, 2020, pp. 1411–1421.

[55] D. R. Silva, "Characterizing relationships between primary miners in ethereum by analyzing on-chain transactions," in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2020, pp. 240–247.

[56] M. Di Angelo and G. Salzer, "Tokens, types, and standards: identification and utilization in ethereum," in *2020 IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*. IEEE, 2020, pp. 1–10.

[57] ——, "Towards the identification of security tokens on ethereum," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021, pp. 1–5.

[58] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the ethereum blockchain," *Expert Systems with Applications*, vol. 150, p. 113318, 2020.

[59] F. Poursafaei, G. B. Hamad, and Z. Zilic, "Detecting malicious ethereum entities via application of machine learning classification," in *2020 2nd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE, 2020, pp. 120–127.

[60] W. Chen, Z. Zheng, J. Cui, E. Ngai, P. Zheng, and Y. Zhou, "Detecting ponzi schemes on ethereum: Towards healthier blockchain technology," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1409–1418.

[61] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: identification, analysis, and impact," *Future Generation Computer Systems*, vol. 102, pp. 259–277, 2020.

[62] E. Jung, M. Le Tilly, A. Gehani, and Y. Ge, "Data mining-based ethereum fraud detection," in *2019 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 2019, pp. 266–273.

[63] C. F. Torres, M. Steichen *et al.*, "The art of the scam: Demystifying honeypots in ethereum smart contracts," in *28th {USENIX} security symposium ({USENIX} security 19)*, 2019, pp. 1591–1607.

[64] W. Chen, X. Guo, Z. Chen, Z. Zheng, Y. Lu, and Y. Li, "Honeypot contract risk warning on ethereum smart contracts," in *2020 IEEE International Conference on Joint Cloud Computing*. IEEE, 2020, pp. 1–8.

[65] K. Qin, L. Zhou, B. Livshits, and A. Gervais, "Attacking the defi ecosystem with flash loans for fun and profit," *arXiv preprint arXiv:2003.03810*, 2020.

[66] J. Wu, Q. Yuan, D. Lin, W. You, W. Chen, C. Chen, and Z. Zheng, "Who are the phishers? phishing scam detection on ethereum via network embedding," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2020.

[67] S. Farrugia, J. Ellul, and G. Azzopardi, "Detection of illicit accounts over the ethereum blockchain," *Expert systems with applications*, vol. 150, p. 113318, 2020.

[68] B. Gao, H. Wang, P. Xia, S. Wu, Y. Zhou, X. Luo, and G. Tyson, "Tracking counterfeit cryptocurrency end-to-end," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 4, no. 3, Nov. 2020. [Online]. Available: https://doi.org/10.1145/3428335

[69] P. Xia, B. Gao, W. Su, Z. Yu, X. Luo, C. Zhang, X. Xiao, G. Xu *et al.*, "Demystifying scam tokens on uniswap decentralized exchange," *arXiv preprint arXiv:2109.00229*, 2021.

[70] O. Dictionary. (2021, 9) Meaning of pull the rug out from under in english. [Online]. Available: https://www.lexico.com/definition/pull_the_rug_out_from_under?locale=en

[71] A. Manuskin, "Unicats go phishing," 10 2020. [Online]. Available: https://medium.com/zengo/unicats-go-phishing-eaf39ff9da64

[72] V. Sopov, "Santadao (hoho) defi pulls exit scam with $200k stolen after being shilled by top twitter influencers," 10 2020. [Online]. Available: https://cryptocomes.com/news/santadao-hoho-defi-pulls-exit-scam-with-200k-stolen-after-being-shilled-by-top-twitter

[73] M. Hunter, "Dxsale to offer airdrop for hatch exit scam victims," 09 2020. [Online]. Available: https://fullycrypto.com/dxsale-to-offer-airdrop-for-hatch-exit-scam-victims

[74] R. MASHAYEKHI, "Worst day in a decade: Nasdaq, sp, dow down nearly 8% in massive market rout," 03 2020. [Online]. Available: https://fortune.com/2020/03/09/dow-jones-sp-500-today-nasdaq-stock-market-crash-2020/s

[75] L. Derczynski, "Complementarity, f-score, and nlp evaluation," in *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 2016, pp. 261–266.

[76] G. Associates, "The rise, fall and rise of sushiswap," 11 2020. [Online]. Available: https://medium.com/gains-associates/the-rise-fall-and-rise-of-sushiswap-ca7753da067f

[77] TMod_Marco, "Requirements how to claim your 400 uni," 9 2020. [Online]. Available: https://gov.uniswap.org/t/learn-requirements-how-to-claim-your-400-uni/1025

[78] L. Aratani, "Squid game cryptocurrency collapses in apparent scam," 1 2021. [Online]. Available: https://www.theguardian.com/technology/2021/nov/01/squid-game-cryptocurrency-scam-fears-investors

[79] SEC, "Sec charges decentralized finance lender and top executives for raising $30 million through fraudulent offerings," 8 2021. [Online]. Available: https://www.sec.gov/news/press-release/2021-145#.YQ2KDMaDtRc.twitter

[80] M. Sigalos, "Why some cyber criminals are ditching bitcoin for a cryptocurrency called monero," 6 2021. [Online]. Available: https://www.cnbc.com/2021/06/13/what-is-monero-new-cryptocurrency-of-choice-for-cyber-criminals.html

[81] N. E. Vincent and A. M. Wilkins, "Challenges when auditing cryptocurrencies," *Current Issues in Auditing*, vol. 14, no. 1, pp. A46–A58, 2020.

[82] R. News, "Leaderboard exploited smart contracts," 2021. [Online]. Available: https://rekt.news/leaderboard/

[83] E. David, "Nexo finance accused of being behind zeus capital and chainlink short," 7 2021. [Online]. Available: https://cointelegraph.com/news/nexo-finance-accused-of-being-behind-zeus-capital-and-chainlink-short

[84] Zach, "Paid influencers," 2021. [Online]. Available: https://twitter.com/zachxbt/status/1443583283648872464

[85] J. R. Searle, "What is an institution?" *Journal of institutional economics*, vol. 1, no. 1, pp. 1–22, 2005.

[86] O. Goldreich and Y. Oren, "Definitions and properties of zero-knowledge proof systems," *Journal of Cryptology*, vol. 7, no. 1, pp. 1–32, 1994.

[87] M. I. Mehar, C. L. Shier, A. Giambattista, E. Gong, G. Fletcher, R. Sanayhie, H. M. Kim, and M. Laskowski, "Understanding a revolutionary and flawed grand experiment in blockchain: the dao attack," *Journal of Cases on Information Technology (JCIT)*, vol. 21, no. 1, pp. 19–32, 2019.

[88] "Smart contract costs – development, audits, transactions," 8 2021. [Online]. Available: https://www.fintechnews.org/smart-contract-costs-development-audits-transactions/#:~:text=Smart%20contract%20audits%20costs,even%20higher%20in%20some%20cases.

[89] A.-D. Popescu, "Transitions and concepts within decentralized finance (defi) space," *Research Terminals In The Social Sciences*, 2020.