FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

# Monero Cross-Chain Traceability

## Empirical Analysis of Privacy Implications from Currency Hard-Forks

DIPLOMARBEIT

zur Erlangung des akademischen Grades

**Diplom-Ingenieur**

im Rahmen des Studiums

**Computational Intelligence**

eingereicht von

**Abraham Hinteregger, BSc MSc**
Matrikelnummer 01025914

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Mitwirkung: Dr. Bernhard Haslhofer

Wien, 19. September 2018

Abraham Hinteregger | Günther Raidl

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

# Monero Cross-Chain Traceability

## Empirical Analysis of Privacy Implications from Currency Hard-Forks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

**Diplom-Ingenieur**

in

**Computational Intelligence**

by

**Abraham Hinteregger, BSc MSc**
Registration Number 01025914

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Günther Raidl
Assistance: Dr. Bernhard Haslhofer

Vienna, 19th September, 2018

Abraham Hinteregger | Günther Raidl

# Erklärung zur Verfassung der Arbeit

Abraham Hinteregger, BSc MSc
Harmoniegasse 1/4, 1090 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 19. September 2018

Abraham Hinteregger

# Acknowledgements

At first, I would like to thank my advisors, Prof. Günther Raidl and Dr. Bernhard Haslhofer for making this work possible by offering their advice and constructive guidance. Furthermore, I would like to express my gratitude to my colleagues at the Austrian Institute of Technology, for their help, organizational and technical support, interesting conversations as well as for introducing me to the intricacies of the "Italian Way of Life". I would also like to thank Christian Feuersänger for making TikZ as great as it is. Finally, I would like to thank Nina and my family for their support, advice and greatness in general.

# Kurzfassung

Bitcoin wurde anfänglich als sichere Methode für anonyme internationale Überweisungen wahrgenommen. Im Laufe der Zeit wurden jedoch Methoden entwickelt, die durch Analyse der öffentlich einsehbaren Transaktionshistorie eine wesentliche Einschränkung der Privatsphäre der Marktteilnehmer ermöglichten. Als Reaktion darauf kam es zur Entwicklung von alternativen Kryptowährungen mit einem verstärkten Fokus auf Privatsphäre und Sicherheit, von denen Monero der Vertreter mit der höchsten Marktkapitalisierung ist. Anfang 2017 haben mehrere Forscher in zwei unabhängigen Publikationen die Unverfolgbarkeit von Monero Transaktionen empirisch analysiert und Methoden und Heuristiken vorgestellt, mit denen für einen Großteil der *Geld*flüsse bis zu diesem Zeitpunkt die Quelle eindeutig bestimmt werden konnte. Infolgedessen wurden mehrere Änderungen vorgenommen um die Unverfolgbarkeit von Transaktionen weiterhin zu gewährleisten. Der Beitrag dieser Arbeit besteht aus zwei Punkten: Wir entwickeln eine neue Methode für die Bestimmung der Quelle von Geldflüssen, die Informationen von der Transaktionshistorie von Forks der ursprünglichen Währung verwendet. Wir verwenden diese Methode um den Einfluss auf die Unverfolgbarkeit von zwei Forks im Frühling 2018 (MoneroV & Monero Original) zu quantifizieren. Weiters untersuchen wir, ob die Maßnahmen gegen bisher veröffentlichten Heuristiken ihr Ziel erreichen. Die Verwendung unserer Methode ermöglicht es uns, für 73 321 Transaktionsinputs (statt für 25 256) im Zeitraum von April bis August 2018 die Quelle zu identifizieren. Wir vergleichen die Ergebnisse dieser Analyse mit den Resultaten der Heuristiken um deren Genauigkeit abzuschätzen. Wir finden, dass die Gegenmaßnahmen ihr gewünschtes Ziel erreicht haben und die Treffsicherheit der Heuristiken maßgeblich beeinträchtigt wurde. Da die Unverfolgbarkeit für einen sehr großen Teil der Transaktionen zum derzeitigen Standpunkt gegeben ist, bleiben Verfahren mit denen andere Kryptowährungen erfolgreich analysiert werden konnten für Monero nach wie vor nicht anwendbar.

# Abstract

While Bitcoin has been initially hailed as a method to anonymously exchange money, researchers made significant strides in analyzing its public transaction history. This lead to the development of cryptocurrencies with a focus on privacy. Monero (based on the CryptoNote protocol) is currently the privacy-coin with the highest market-capitalization. In 2017, Monero's untraceability guarantees were analyzed in two independent publications. They found that a majority of the transaction made up to that point were traceable, raising some doubts concerning the veracity of the Monero's privacy claims. Following these publications several improvements to Monero's protocol have been rolled out. The contribution of this work is twofold: First, we introduce an additional tracing method, which is based on currency hard forks and apply it to measure the privacy-loss stemming from two Monero forks in spring 2018 (MoneroV & Monero Original) Additionally, we evaluate the effective of the countermeasures by analyzing the accuracy of the heuristics employed in previous publications on a recent export of the Monero blockchain. We find that our additional method enables us to trace 73 321 transaction inputs (up from 25 256 using established methods) in the time-frame between April and August 2018. Based on the inputs traced with existing as well as our new method, we estimate the accuracy of previously proposed heuristics. Our results suggest that the countermeasures were effective and reduced the accuracy of the heuristics significantly. As most of the transactions remain untraceable, methods applied for the analysis of other cryptocurrencies are still not applicable to Monero's transaction data.

# Contents

# Part I

# Cryptocurrency Analytics: Theoretical Background

CHAPTER 1

# Introduction

In 2008, Satoshi Nakamoto published his whitepaper on Bitcoin, a digital currency which uses a decentralized consensus protocol (Nakamoto Consensus) to establish a global storage with decentralized validation, the so-called Blockchain, where the Bitcoin transaction history is stored. Initially, it seemed like Bitcoin enabled private, anonymous transactions and has been adopted for various black markets. Since then, Blockchain analysis techniques such as the multiple input heuristic and change heuristics ([Reid and Harrigan, 2013, Androulaki et al., 2013, Nick, Jonas David, 2015, Haslhofer et al., 2016]...) have been developed which allowed to identify real-world entities behind sets of addresses, annulling the privacy mechanism of Bitcoin. These shortcomings have led to the developments of alternative cryptocurrencies with a stronger focus on privacy, such as Monero. Monero's aim is to employ cryptographic methods to provide a private, untraceable and fungible cryptocurrency. This is accomplished with different strategies, such as concealed transaction amounts (confidential transactions), hiding of sender and recipient addresses (stealth addresses) and obscuration of the transaction graph (ring signatures, [Noether et al., 2016]). Nevertheless, analysis techniques have been proposed by [Kumar et al., 2017] and [Möser et al., 2018], which reveal parts of the transaction graph. This is accomplished via identification of known decoys the ring signatures and by exploiting differences between patterns arising from user behavior and from the rather simple decoy sampling techniques. Since then, Monero's developers made changes (more members for ring signatures and better sampling techniques) to address these vulnerabilities.

In the second part of this work we will try to answer the following questions:

- What was the privacy impact from two Monero hard forks (MoneroV, Monero Original) in spring 2018?
- Were the improvements to the protocol aimed at the tracing-heuristics used by [Kumar et al., 2017] and [Möser et al., 2018] effective?

For this purpose we perform a traceability analysis (extended with our newly proposed method for analyzing currency hard forks) on a recent export (August 2018) of the Monero blockchain. We will quantify the privacy impact by analyzing the transaction data of Monero, MoneroV and Monero Original in the months following the hard forks. Based on the data obtained in our traceability analysis we will estimate the performance of the tracing-heuristics on recent transactions.

In Chapter 2, we will explain how Bitcoin, the blockchain and the underlying P2P network function and operate. Subsequently, we will introduce the key features of Monero, a cryptocurrency based on the CryptoNote protocol.

In Chapter 3 we will present the state of the art in blockchain analysis for Bitcoin and the applicability of these methods to Monero, followed by an overview of the techniques specific to Monero in Chapter 4.

In Chapter 5 we propose a new Monero-specific analysis technique, which exploits information gains from currency hard forks. This is an attack vector that the Monero community has already been aware of, but we're not aware of any previous work that analyzed its impact.

In Chapter 6 we present the results of our traceability analysis. The ground-truth data obtained in this analysis is then used to evaluate the performance of the heuristics proposed by [Kumar et al., 2017] and [Möser et al., 2018] for recent transactions, the results of which will be presented in chapter Chapter 7. Additionally, we will expand slightly on the analysis found from [Wijaya et al., 2018].

In the last and final chapter we will then summarize and discuss our results.

## Other Contributions and Reproducibility

The toolchain which has been developed for this analysis is released under MIT license on GitHub[1]. All graphs and statistics contained in this work can be reproduced by following the steps of the provided README. For this purpose, all plots reference the query and the CSV files on which they are based.

As Monero's untraceability is based on the sampling of decoy inputs for each real transaction input, it is important that the decoys are not references to outputs which are known to be spent, as they would not contribute to the size of the anonymity set of the real input. To prevent the sampling of spent outputs as decoys, the blackballing tool has been released together with Monero v.0.12, though most users lack a database of known spent outputs. To address this, we publish a list of known spent transaction outputs and outputs which are at risk of being identified as spent (see Section 6.4).

[1] [Hinteregger, 2018b]: `https://github.com/oerpli/MONitERO`

CHAPTER 2

# Cryptocurrencies

In 2008, Satoshi Nakamoto published a whitepaper [Nakamoto, 2008] that introduced Bitcoin, the first decentralized digital currency. In the whitepaper, Nakamoto introduced a distributed ledger called Blockchain, which solved the problem of double-spending without a central authority. This was combined with a proof of work algorithm known from Hashcash ([Back, 2002]), where the goal of the method was to combat spam by adding a computationally intensive workload to emails (resulting in a cost, similar to stamps) which could be verified efficiently (i.e. cheaply).

While the blockchain technology delivered on establishing a shared consensus and trust in the currency, its public nature enabled cryptocurrency analytics which lead to doubts concerning the privacy offered via its pseudonymous addresses. Additionally, during several boom (and bust) cycles, the scalability of the currency has been tried, tested and found out to be in need of improvement.

Over the years, various alternative-currency projects (sometimes referred to as altcoins) were introduced, usually with the goal to address a specific or multiple shortcoming of existing offerings, such as:

- Privacy concerns (Monero, ZCash)
- High transaction fees (Litecoin)
- Environmental impact of mining (Gridcoin)
- No dog on the logo (Dogecoin)

In this chapter, we will first explain the key features of Bitcoin to introduce the most important shared concepts of cryptocurrencies. This is followed with a section about

Monero, a cryptocurrency based on the privacy-oriented CryptoNote protocol. There we will highlight its features and key differences to the baseline, set by Bitcoin.

Even though these cryptocurrencies use cryptographic methods (thus the name) extensively to provide trust and privacy, we do not currently know about efforts to use cryptographic approaches for linking or tracing transactions. The cryptographic methods are therefore presented briefly, but proving correctness of the approach and implementation is out of the scope of this work and therefore assumed. Finding eventual flaws is left as an exercise to the reader.

## 2.1 Bitcoin

As Bitcoin was the first cryptocurrency of its kind we will first give a brief overview how the different parts work in its implementation. Basically, there are two components which work together to ensure the operational capability of Bitcoin:

- A peer to peer (P2P) network of nodes which exchange packets of (transaction-) data and ensure their integrity.
- An append-only data-storage system—called *Blockchain*—that ensures decentralized consensus and trust, a necessity for a currency.

### 2.1.1 Peer to peer network

The actors participating in the decentralized Bitcoin network are called *nodes*. Each node connects to some subset of the other nodes and exchanges messages with them. Some of the message types are:

- *version* & *verack*: Used when connecting to peers. Node A sends a message with its version to another node B. This peer then responds with *verack* if it accepts connections from clients with that version.
- *inv*: If the node gains knowledge of a new block/transaction it sends out a list of transactions/blocks.
- *getdata*: If a node gets relayed a transaction or block it does not yet know about via *inv* from a peer, it requests the data of the block with a *getdata*-message. If it considers the transaction/block valid it propagates the existence of the transaction/block to its peers (with *inv*).
- *tx* & *block*: Send a transaction or block as response to a *getdata* message.

While all nodes that participate in the network use the messaging protocol to receive or relay transactions and blocks, only a subset of the nodes, the so-called miners, also try to find new blocks on their own.

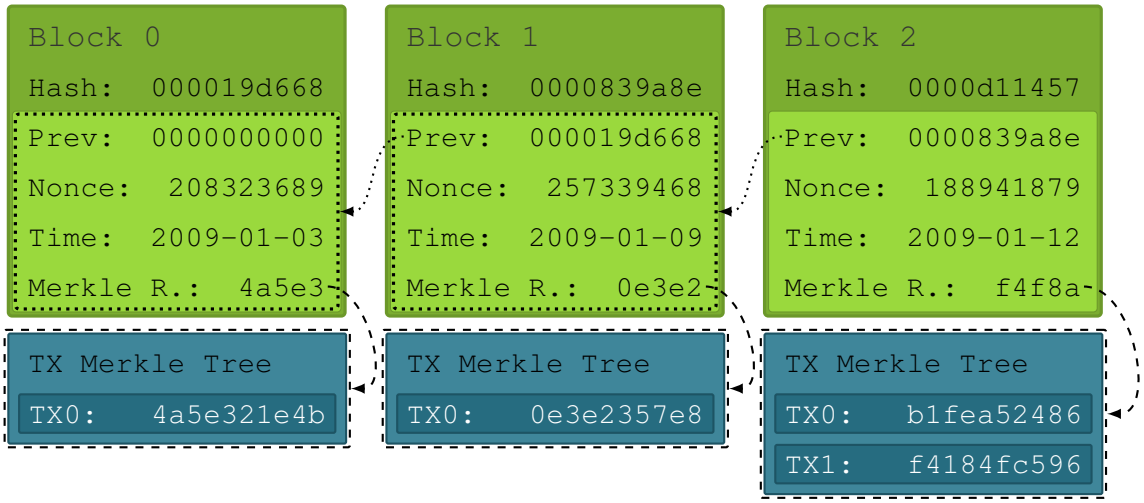


Figure 2.1: **Blockchain:** Blockchain with genesis block and the first two blocks. Each block consists of a header which references the root of a Merkle tree (dashed) that contains the transactions included in the block. Each block has a coinbase transaction (TX0). In this example, block 2 is the first block with a "real" transaction. Each block references its preceding block with a hash of its header (the dotted part).

### 2.1.2 Blocks & Blockchain

The *blockchain* is a decentralized data structure that stores the state (i.e. current currency distribution) by storing all transactions that happened until now. It consists of *blocks* (see Figure 2.1), each of which consists of the header and a set of transactions.

#### 2.1.2.1 Block Header

The header contains the following data:

- Version: This number specifies the validation rules that the block adheres to.
- Timestamp: The time when the miner started mining this block. As miners operate on the whole world and their communication may have some latency, timestamps of consecutive blocks are not ordered. Each block must have a timestamp that is higher than the timestamp of the preceding 11 blocks though.
- Difficulty: New blocks are accepted, when the hash of their header is below a certain threshold. This threshold is saved in encoded form on the block and is based on the hashing power of the network and is updated every 2016 blocks.
- Nonce: As the other fields of the header are determined by other factors, miners vary the value of this field until the hash of the block header passes the target threshold.

- Merkle root of transaction hashes: The transactions included in the block are referenced in the leaves of a Merkle tree (see Section 2.1.2.2). The hash of the root node (called *Merkle root*) depends on all the transactions in the tree as well as on their order and (ignoring collisions) is uniquely determined by them and is included in the block header. This ensures that the included transactions cannot be modified after the block has been accepted.

- Hash of the previous block header: Establishes a linear order from the most recent block back to the Genesis block[2], ensuring that none of the previous blocks have been tampered with, as this would invalidate the reference.

If a node would try to send a modified (and thus invalid) block to its peers, those peers would not relay it further as they would notice the integrity violation.

#### 2.1.2.2 Transactions

It is critical that the transaction cannot be altered after they've been added to the blockchain (else it would be possible to modify account balances by e.g. swapping out transaction outputs). Additionally, it should be computationally inexpensive to check whether a specific transaction is in a block. For this purpose, Merkle trees (also called hash trees) are used. A *Merkle tree* is a binary tree where data is stored in the leaves and each node is labelled with a hash that is either derived from the data (in case of leaf node) or from the hash values of its two direct ancestors. This allows adding verifying transactions in $\mathcal{O}(\log n)$ time and enables efficient verification of transactions, as not the whole tree but only the path from root to the transaction of interest as well as the hashes of the other branches must be synchronized.

### 2.1.3 Mining

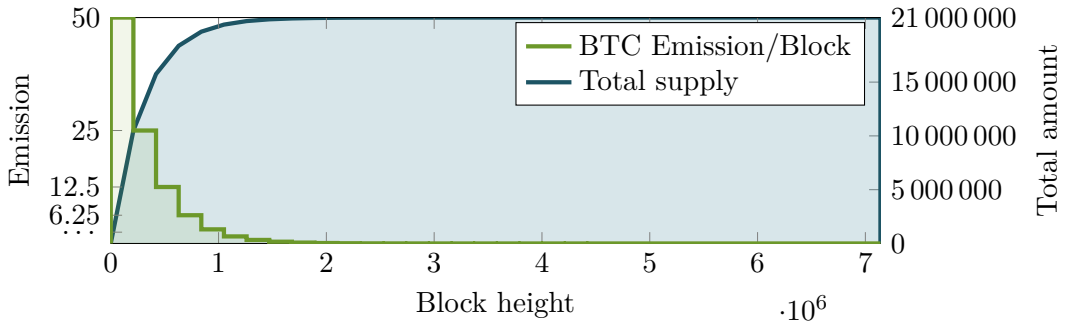


Figure 2.2: **Bitcoin Emission:** For Bitcoin, emission per block is halved every 210 000 blocks and will reach 0 at block 6 930 000. Total supply at that point is 20 999 999 BTC.

[2] https://blockchain.info/de/block-height/0

Each block has a unique hash that is generated by applying a hash function to its header. This hash must fulfill a difficulty criterion, i.e. it must be lower than some value[3]. For this purpose, a field called *nonce* is included in the block header which can be varied until the hash of the block header fulfills the requirements. If a miner finds a nonce that results in the block header fulfilling the difficulty criterion, it publishes the existence of this new valid block to its peers (see section 2.1.1).
Nodes are incentivized to expend resources to find valid blocks by two mechanisms:

- Transaction fees: The fees assigned by Bitcoin users to transactions, which are included in a block, are added to the block reward.
- Block reward: Each block contains a transaction, called *coinbase transaction* to an address that can be provided by the miner (usually their own address). Initially this reward has been 50 BTC but halves every 210 000 blocks. Due to this there is an upper bound of 20 999 999.9769 BTC in circulation, calculated as follows[4]:

$$\frac{\sum_{i=0}^{32} 210\,000 \left\lfloor \frac{50 \cdot 10^8}{2^i} \right\rfloor}{10^8}$$

  Both, the emission rate and the total supply at a given block height can be seen in Figure 2.2.

If at some point all bitcoins have been mined, the transaction fees remain the only incentive for miners to continue finding new blocks.

### 2.1.4 Transactions

Bitcoins are transferred from one actor to another via transactions. Each transaction consists of one or more inputs (except coinbase transactions) and one or more outputs, as can be seen in Figure 2.3. Each input of a transaction is a reference to a yet unspent output of another transaction (abbreviated as UTXO) as well as a signature as proof of ownership (to make sure that people don't spend assets that don't belong to them). Outputs consists of two parts: the desired amount and a script which contains the hash of the public address of the recipient. When an output is referenced in another transaction, the sender has to make sure that output-script combined with the signature he provides evaluates to true and thus confirms that he's authorized to use the associated coins. After this, the referenced output is spent and cannot be used as input in another transaction. The sum of the amounts associated with the inputs must be at least the sum of the amounts of the outputs, the (positive) difference between the two sums is the fee, a reward for the miner of the block that contains the transaction. There are ways to create transactions that provide different methods of verifying ownership, but these are uncommon and not relevant for this work.

[3] A higher difficulty corresponds to a higher amount of leading zeros in the hash.
[4] The factor $10^8$ is to calculate from BTC to Satoshi and back again.

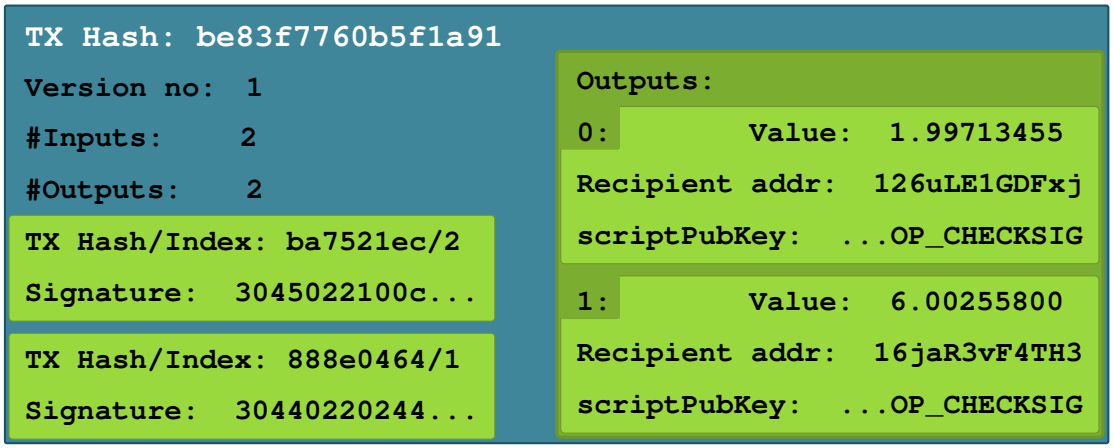


Figure 2.3: **Schema of a BTC transaction:** Each transaction has some number of inputs and outputs. Inputs are references to TX outputs (TX hash & index of output) and for each input a signature is provided, which is used to prove that the input signer is authorized to spend the referenced output. Each output has an index, a value, a recipient address and a scriptPubKey, which is used to verify the signature provided by the spender.

### 2.1.5 Blockchain Splits & Forks

If run on a single node, the blockchain would be similar to a (reversed) linked list, where each block points to its predecessor. In practice, the blockchain may look like a tree, where at some points multiple elements may have the same predecessor. These splits may happen due to different reasons and are classified into the following categories: Blockchain forks, hard forks and soft forks.

#### 2.1.5.1 Blockchain fork

A *chain fork* happens when two miners find a new valid block on the same blockchain height at almost the same time[5]. After a chain split occurs, the following mechanism ensures that after some time, the two branches of the blockchain are combined to a single blockchain again[6]. Per definition, if a fork occurs and there are two competing chains, the one with the higher sum of difficulty ("the longer chain") is the valid one (in case of equal total difficulty, the one that was received earlier is preferred). Miners that get relayed two candidate chains are incentivized to mine on the valid chain, as transactions (including the reward for mining) on the stale chain are lost. In Figure 2.4 a blockchain fork with an orphaned fork of length 2 is illustrated.

[5] If the first of the two blocks is found at time $t_1$ and the second at time $t_2$, where $t_1 < t_2$, the difference $\Delta t = t_2 - t_1$ is smaller than the time needed for the first block to be propagated in the P2P network to the second miner.

[6] Assuming that more than half of the mining power is not compromised.

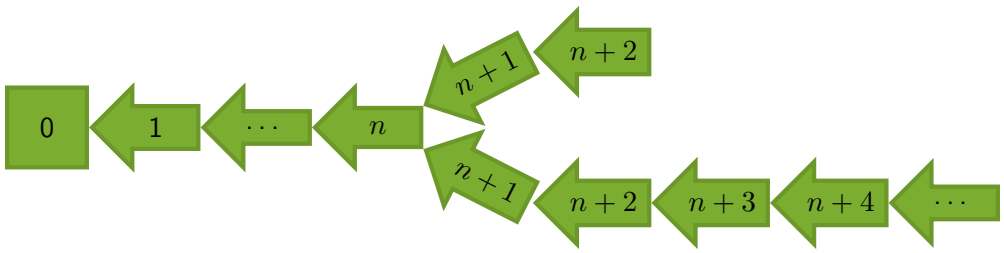


Figure 2.4: **Blockchain fork:** At blockchain height $n$ two new blocks are found in a short time span and thus two blocks at height $n + 1$ exist. One chain progresses faster and thus gets continued, the two blocks on the other chain become orphaned blocks and any transactions included in them that are not yet part of the main-chain are added to a block of the main-chain.

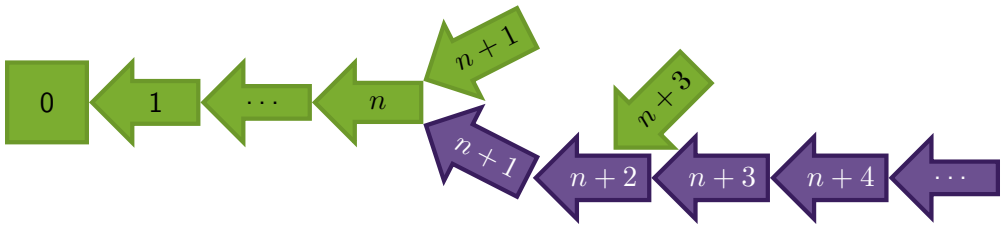


Figure 2.5: **Soft fork:** Up to block height $n$, all blocks are mined with the old rules ( ⬅ ). Then, a majority of nodes activate the new client version which rejects blocks issued with earlier versions. Clients with the old version accept blocks which adhere to the new rules ( ⬅ ) and may mine some valid blocks ($n + 1$ and $n + 3$ in the image), though these blocks are quickly orphaned, as the majority of miners reject them and the chain with the updated rules ( ⬅ ) progresses faster.

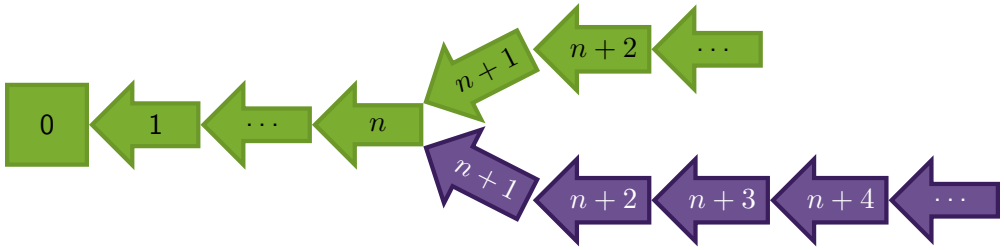


Figure 2.6: **Hard fork:** Up to block height $n$, all blocks are mined with the old rules ( ⬅ ). Then, the block rules from the hard fork are activated on nodes with the new client version and nodes using the pre-fork rules reject the new blocks ( ⬅ ). If a majority of the miners uses the updated client, the blockchain with the updated rules progresses faster. Clients using the old version may choose to continue mining on the old blockchain.

#### 2.1.5.2 Hard & Soft forks

A *soft fork* is a change to the protocol that introduces additional restrictions, e.g. to the maximum size of the block or the transaction format. If not all nodes switch to the version of the software that implements the soft fork, there are two competing protocols at the same time: the legacy version and the soft-forked version. While all nodes recognize blocks mined with the soft-forked version as valid blocks, blocks mined with the legacy version may be rejected by nodes running the soft-forked software. This is illustrated in Figure 2.5. A well-known example of a feature rolled out with a soft fork is SegWit (short for Segregated Witnesses) which has been activated on August 24$^{\text{th}}$, 2017.

A *hard fork* loosens some restrictions, which results in legacy blocks being compatible with the hard forked blockchain but not the other way around. If a majority of users upgrade, the hard forked blockchain progresses faster. If some users choose to continue mining with the old rules, the old blockchain can be continued. A case where the majority of users upgrade to the new client is illustrated in Figure 2.6. Depending on the split of users between the versions of the software, both soft and hard forks may pan out differently[7]. A well-known example of a Bitcoin hard fork is the introduction of Bitcoin cash, which increased the maximum block size from 1MB to 8MB.

#### 2.1.5.3 Alternative history attacks

A transaction in a blockchain can only be valid if the outputs that are spent in the transaction have not been spent before. To ensure integrity of the blockchain, nodes verify that a transaction does not try to spend outputs which already have been spent, before they include the transaction in the blockchain. For this verification, all transactions up to the current block height as well as transactions already included in the current block are considered.

A malicious user (MU) could try to double spend some output by issuing two transactions: one which transfers some amount to a merchant and another which invalidates the first transaction (if it reaches the node which mines the next block first). To prevent this very simple exploit, users usually wait until a transaction has been confirmed in the blockchain as valid, before accepting it. If a MU has considerable hashing power at its disposal, they could issue the transaction and then mine several blocks with an additional unpublished transaction that invalidates the transfer to the merchant. After the merchant has waited for the transfer to be confirmed (by being included in the blockchain) the MU publishes the alternative branch of the blockchain, which, if it is longer than the other branch at that point, is considered the valid chain, which invalidates and rolls back the transfer to the merchant. As a safeguard against this kind of attack, it is usually advised to not only wait until a transaction is included in the blockchain, but also to wait for $n$ (with $n = 6$ being a common choice) *confirmations*, which means that if a transaction has been included in some block, $n$ blocks have been added to the same branch of the blockchain. If a MU controls more than 51% (or even less, see [Eyal and Sirer, 2014]) of the hash

[7] More details can be found at [Light, 2017].

rate of the network, such an attack cannot be prevented, even with a high confirmation threshold for accepting transfers.

#### 2.1.5.4 Airdrop

Altcoin developers may want to accelerate adoption of their currency by distributing it among some active users to gain mindshare and free advertising. This can be accomplished by hard forking the blockchain of an existing currency at some arbitrarily chosen height. Then, all users that have unspent outputs from pre-fork transactions can spend those outputs on both blockchains, as nodes on each chain only verify if the outputs have not yet been spent on their blockchain, which consists of the original blockchain up to the fork and either its direct continuation or the forked chain, but not both. The funds that owners of the original currency gain due to this is referred to as *airdrop*.

## 2.2 Monero

Monero is a cryptocurrency with heavy focus on privacy, based on the CryptoNote protocol introduced in [Van Saberhagen, 2013]. Its aim is to improve upon Bitcoin in the following areas:

- Linkability: As the recipient address of each Bitcoin transaction output is known, it is possible to infer that two transactions that spend outputs belonging to the same address were issued by the same user.
- Traceability: As it is possible to follow the trace of a bitcoin via its transactions, it is also possible to determine if a coin has been involved in "suspicious activity". Exchanges could blacklist funds that are flagged in this way.
- Limited supply: There are doubts concerning the viability of the mining market, when the block reward is comprised of only transactions fees, as the incentive structure could lead to large mining syndicates creating alternative histories if they miss transactions with a large block reward[8].

Linkability and Traceability are addressed with cryptographic methods, which hide the real transaction inputs (untraceable transactions), recipient addresses (unlinkable transactions) and amounts (confidential transactions). The first is realized with adding decoys to each output spent in a transaction, the second with using one-time public keys instead of addresses and the third with confidential transactions, called *RingCT*.

The "problem"[9] with limited supply is solved with a lower bound on the block reward.

The following subsections provide more details on these methods.

[8] More details can be found at `http://weuse.cash/transparent-emission/`.

[9] We are not aware of a consensus whether a limited supply is beneficial or not. See `https://bitcointalk.org/index.php?topic=753252.msg12440450#msg12440450` for some arguments.

### 2.2.1 Ring Signatures: Untraceable Transactions

A Bitcoin transaction has one or more TXOs as input which are spent after they've been used once. It is therefore possible to trace a bitcoin from its origin (as output of a coinbase transaction) to its current address by following the transaction history. In contrast, each Monero transaction input references a set of possible TXOs, of which only one is spent in the transaction. Furthermore, it is (in theory) very hard for anyone except the creator of the transaction to identify the TXO that is really used in the transaction. For this purpose, *one-time ring signatures* and *key images* are used. Key images are uniquely determined for each transaction output and are used to prevent double spending (each key image may only occur once on the blockchain). The ring signature ensures that the key image provided by the issuer of the transaction is in fact generated from the private key of one of the referenced outputs. This is accomplished with a protocol, based on the traceable ring signatures introduced in [Fujisaki and Suzuki, 2007]. Therefore, (usually) only the creator of the transaction can identify the TXO which has been spent. While beneficial for privacy, this leads to some technical difficulties: the blockchain cannot be pruned as it is usually not possible to identify outputs that have been verifiable spent and can thus be removed (outputs can be referenced in Bitcoin transactions only once, whereas Monero outputs can be referenced arbitrarily often).

### 2.2.2 Stealth Addresses: Unlinkable Transactions

Each Bitcoin address is an unambiguous identifier for incoming transactions. Whenever two separate transactions are sent to the same address, it is possible to say with certainty that both transactions have the same recipient. If a Bitcoin user does not want separate transactions to be linkable, they must provide unique destination addresses for each transaction and be careful not to spend outputs which have been sent to separate addresses in a common transaction, as this would reveal that the two addresses are owned by the same entity (see Section 3.1).

Monero solves this problem by using unique destination keys that are derived from the recipients address and random data from the sender. These destination keys are generated as follows (Bobs private key is $(a, b)$ and his public key is $(A, B) = (aG, bG)$ where $G$ is the base point of the elliptic curve[10]):

1. Alice wants to pay Bob and uses his public key $(A, B)$.
2. Alice generates a random number $r$ and computes a one-time public key $P = \mathcal{H}_s(rA)G + B$ (where $\mathcal{H}_s$ is a hash function), which is used as destination key for the transaction
3. Alice adds the value $R = rG$ to the transaction[11]. $R$ is used for the Diffie-Hellman-Merkle key exchange ([Diffie and Hellman, 1976]).

[10] For more details see [Hankerson et al., 2006]

[11] As the random value $r$ can be reused for multiple destination keys $P_i$, only one $R$ must be included in a transaction, even if there are multiple outputs.

Bob is then able to redeem the funds sent to him by using his private key $(a, b)$ and applying the following steps:

1. For every passing transaction:
   a) Bob uses the $R$ included in the transaction to calculate $P' = \mathcal{H}_s(aR)G + B = \mathcal{H}_s(a \cdot r \cdot G)G + B$
   b) If Bob is the recipient of the address, $P = P'$ (this follows from $\mathcal{H}_s(rA) = \mathcal{H}_s(r \cdot aG) = \mathcal{H}_s(a \cdot rG)$)
2. Bob then calculates the one-time private key $p = \mathcal{H}_s(aR) + b$. He can than sign a transaction with p and it is possible to verify that the output belongs to him because $P = pG = (\mathcal{H}_s(aR) + b)G = \mathcal{H}_s(aR)G + B$.

As transactions are sent to one-time public keys derived from the public address of the recipient and random data, it is not possible for third parties to determine who the recipient is (without cooperation by either the sender or the recipient, see Section 2.2.2.1). Furthermore, this prevents address reuse, as each TXO is associated with a unique public key.

In practice, there are a few exceptions to this[12] that most likely stem from bugs in some wallet software that are probably fixed by now, as the most recent occurrence of a duplicate public key has been in October 2016.

If two public keys collide, only one of the affected outputs can be spent, as they then also share their private spend key $p$ which may only be used once to prevent double spending. Sending monero to an already existing public key thus burns the associated amount of the output. It is therefore unlikely that a malicious user would deliberately create public key collisions (by issuing multiple transactions with the same random value $r$) as it would result in the loss of the MUs funds without disclosing information about the target of the transaction.

Theoretically, a public key collision could occur by random chance, though it is rather unlikely that this happens[13].

#### 2.2.2.1 View keys

As only one part ($a$) of the private key ($(a, b)$) must be used to identify transactions belonging to a public address, it is possible to share $a$ (the so-called view key) to allow third parties to monitor transactions involving an address. This can be used to comply with regulations or to provide accountability that would otherwise be lost due to the use of stealth addresses.

[12] See `https://git.io/fNLuD` for a complete list of all public keys that occur multiple times on the Monero blockchain and `https://git.io/fNLaU` for a list of all the occurrences.

[13] The problem is analogous to the birthday paradox. [Luigi1111, 2016] calculates that it would take roughly $2^{126} (= \sqrt{\#\text{possible keys}})$ attempts to have a 50% chance of a collision.

### 2.2.3 RingCT Confidential Transactions: Hidden amounts

Confidential transactions hide the denomination of outputs (therefore also inputs) and the total output of transactions. This leads to better privacy in several ways:

1. Only transaction outputs with matching denominations can be used as mixins for a transaction. The set of mixin candidates may therefore be smaller than desired for outputs with a very unusual denomination (some empirical analysis of this can be found in [Kumar et al., 2017]). Hiding the denomination allows to use any other output as mixin, thus solving this problem.

2. If someone would offer goods of questionable legality for some price $c$, law enforcement could look specifically for transaction with that output amount.

3. Some address clustering schemes use heuristics related to input and output sizes to identify addresses or transactions belonging or involving to the same actor(s) (e.g.: [Quesnelle, 2017]). Hiding the denominations removes this as possible attack vector.

### 2.2.4 Infinite Supply

Whereas Bitcoin and most of the cryptocurrencies derived from it only have a finite supply of (slightly below) 21 million coins, Monero has an infinite supply, as the emission per block has a lower bound of 0.6 monero (0.3 monero per minute with 2 minute block time). Due to this lower bound the total amount does not converge, as can be seen in Figure 2.7.

The block reward (excluding transaction fees) $B_n$ in tacoshi[14] for block $i$ is based on $C_i$, the cumulative emission up to block $i$ and can be calculated as follows :

$$C_n = \sum_{i=0}^{n-1} B_i$$

$$B_n = \max\left(0.6 \times 10^{12}, \begin{cases} (2^{64} - 1 - C_n) \times 2^{-20} & \text{for } n \leq 1009827 \\ (2^{64} - 1 - C_n) \times 2^{-19} & \text{for } n > 1009827 \end{cases}\right)$$

The changed exponent for $n > 1009827$ is due to the changed block time (see Section 2.2.5) and is also the reason for the discontinuity in Figure 2.7.

### 2.2.5 Major Changes to the Protocol

Since its initial release, Monero has been continuously developed to improve privacy and additional features. While some changes (for example a different sampling of mixins in

[14] *Tacoshi* is a reference to the smallest possible unit of Bitcoin, called Satoshi. It is named after Monero developer TacoTime. Sometimes it is called *Piconero*, a portmanteau of Monero and the SI-prefix pico for $10^{-12}$.

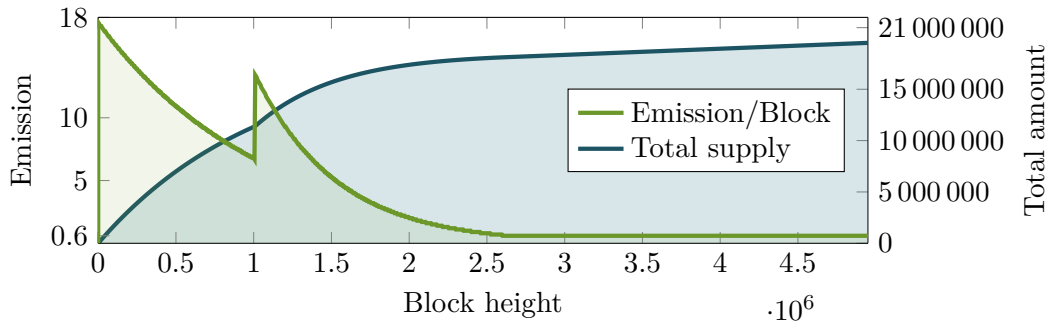


Figure 2.7: **Monero Emission:** For MONERO, emission follows a smooth curve with a lower bound of 0.6. Notice the jump at block height 1 009 827, where the block reward and block time have been doubled. When the minimum block reward has been reached, total supply grows linearly without bound.

transactions) could be rolled out with updated clients and were compatible with nodes running on older versions of the software, other updates were distributed as hard forks and made it necessary to update the client (see Figure 2.6).

In Table 2.1 the major changes with relevance to linking are listed. A dash in the columns *Fork* and *Block #* marks changes that were rolled out via updates to the client, the date is then the release date of the specified client version. Otherwise the fork version and block number from when the fork happened, are provided (empty fields refer to the last value in the same column), in which case the date refers to the timestamp of the block where the fork took place. The changes to the distribution of mixins primarily affect tracing-strategies that employ temporal analysis (e.g. the guess newest heuristic, see Section 4.3). As for this work only deterministic strategies were employed, our methods were unaffected by changes to the mixin-distribution.

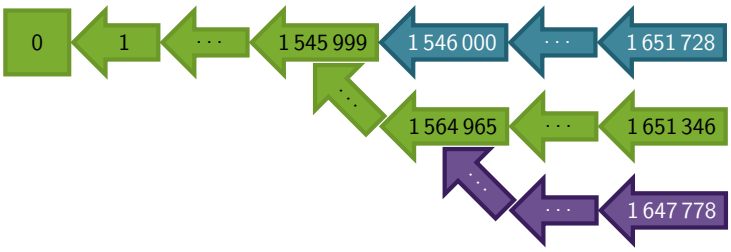


Figure 2.8: **Spring 2018 Monero Forks:** Illustration of the blockchains used for the empirical part this work. The Monero (XMR) blockchain went from the genesis block (0) up to block number 1 546 599, where a scheduled hard fork (to prevent ASIC mining) took place. Disagreement in the community lead to the forked currency Monero Original (XMO), continuing the Monero v6 compatible blockchain. At block height 1 564 965 the currency has been forked again, this time resulting in MoneroV (XMV).

Table 2.1: Table of versions and forks of Monero that affect tracing inputs. Some changes were rolled out with a new client version (mostly improved mixin-sampling methods) and were compatible with older clients. Other changes made a hard fork necessary, in which case the fork version and block are provided.

| Fork | Block # | Date | Client | Changes that affect tracing |
|---|---|---|---|---|
| - | - | 2016-01-01 | v0.9.0 | Mixins sampled from triangular distribution |
| v2 | 1 009 827 | 2016-03-22 | - | Ringsize $\geq 3$ |
| | | | - | block time doubled (from 60s to 120s) |
| - | - | 2016-12-13 | v0.10.1 | Sample $\approx$25% of inputs with age $\leq$5 days |
| v4 | 1 141 317 | 2017-01-05 | - | RingCT enabled |
| - | - | 2017-09-07 | v0.11.0 | More mixins with age $\leq$1.8 days (instead of 5) |
| v6 | 1 400 000 | 2017-09-16 | - | Ringsize $\geq 5$ |
| | | | - | RingCT only |
| - | - | 2018-03-24 | v0.12.0 | Mitigation against key-reuse-attacks from forks |
| v7 | 1 546 000 | 2018-04-06 | - | Ringsize $\geq 7$ |
| | | | - | New Hash algorithm (to prevent ASIC mining) |
| | | | - | Enforce input sorting |

### 2.2.6 Spring 2018 Monero forks

In spring 2018, the Monero blockchain split up into (at least) three separate blockchains. One was the original Monero blockchain, which forked at block height 1 546 600 to introduce the new ASIC resistant hashing algorithm. This lead to Monero Original (note the capitalization), as parts of the Monero community did not agree with the direction and preferred the previous, ASIC-minable algorithm[15]. While this fork initially had some support, due to miners not wanting to lose their investment into special purpose hardware, in the weeks following the forks activity on the network fizzled out and apart from updated logos and names in the readmes, no observable development took place.

In February 2018 MoneroV was announced[16], which forked from the Monero blockchain at block height 1 564 965 (2018-05-03, later than initially planned), though the client and network were only released at height 1 565 244 (on 2 005), resulting in a "pre-mine" of $\approx$5.8%. The team behind MoneroV started their project with the intention of adapting Monero to make it adhere to Austrian economics (i.e.: capping the available supply at 256 000 000 XMV), and promising several improvements and features, which would address some of the shortcomings of Monero.

[15]Technically, there is also Monero Classic and Monero 0, both with the same goal, Monero-client version and blockchain as Monero Original. While we refer to Monero Original in this work, Monero Original/Classic/0 would be more precise.

[16]`https://bitcointalk.org/index.php?topic=2947912.0`

CHAPTER 3

# Bitcoin Analytics Techniques

Since its inception, Bitcoin has gained a reputation as the currency of choice for various dealings of questionable legality (see [Foley et al., 2018, Paquet-Clouston, 2017, Christin, 2013, Gwern Branwen, 2011]). Naturally, privacy was a concern for people involved in illegal activities, though other users may also value privacy, even if they have nothing to hide. Through the use of pseudonymous addresses, Bitcoin facilitates some level of anonymity for its users, though the blockchain as a public record of all transactions allows everyone to link transactions to and from a person, assuming that their address is known.

Therefore, it was often advised to use multiple addresses to counteract this very simple analysis, which then lead to the employment of more advanced approaches for the deanonymization of blockchain based cryptocurrencies. In [ShenTu and Yu, 2015], the different deanonymization-approaches were categorized as follows:

1. Analysis of the P2P Network: It is possible to observe the communication of nodes in the network and analyze messages sent between them to obtain information about e.g. the origin (network address) of a new transaction.

2. Analysis of the Transaction Chain: As all transactions can be obtained from public blockchain data it is possible to infer some knowledge from certain patterns.

3. Eavesdropping: If addresses or transactions are mentioned in public fora or on social media it is possible to connect the author to those addresses or transactions.[17]

For all three approaches some techniques exist to impede deanonymization to some extent, e.g.:

[17] Famous case: Silkroad `https://en.wikipedia.org/wiki/Ross_Ulbricht`

1. Employing alternative routing technologies, such as the TOR network, The Invisible Internet (I2P) and Transaction Remote Release (TRR) is a possible way to prevent the first kind of attack.

2. Coin-mixing services or Coin-Joins (see section 3.1) allow users to hide the origin of their coins, which makes following the chain of transactions harder.

3. Regularly switching addresses allows adversaries to only link a subset of the transactions a person has been involved in to his real identity.

Nevertheless, the public nature of the Bitcoin transaction history via its blockchain limits the privacy of its users. This lead to the development of privacy-minded altcoins, such as Monero or ZCash, with the goal of bringing real anonymity to cryptocurrency transactions.

As this work is focused on deanonymization via analysis of the transaction chain, we will first provide an overview of the available approaches for Bitcoin. For these methods, their applicability to Monero is also explored. We will not cover methods that exploit analysis of the P2P network (e.g. [Biryukov et al., 2014]), or the bloom filter vulnerability presented in [Nick, Jonas David, 2015]. We also will not go into methods that combine data from sources other than transaction-data on the blockchain ([Haslhofer et al., 2016, Goldfeder et al., 2017, Ermilov et al., 2017]). This is followed with deanonymization methods aimed specifically at Monero, followed by a chapter dedicated to a new method we propose and analyze subsequently.

As each Bitcoin transaction—including involved addresses and amounts—is recorded on the public blockchain, it is possible to build the *address graph*, where nodes represent addresses and weighted arcs represent the coins going from one address to another in a transaction. As Bitcoin users may have multiple addresses, several nodes in the network may represent the same user. It would therefore be of interest to find all the addresses belonging to the same user and combine them into clusters, where each cluster represents a real-world entity. This graph, derived from the address graph, is called the *entity graph* (see e.g. [Reid and Harrigan, 2013, Haslhofer et al., 2016]). This is illustrated in Figure 3.1.

For this purpose, clusters of addresses in the *address graph* which belong to the same user must be found. Sets of addresses belonging to the same user could be provided by the users themselves, e.g. via tagging services[18], though this would help only in a minority of cases. Therefore, several techniques have been developed which use different technical means (see enumeration in Chapter 3) to produce sets of addresses belonging to a user, though we only focus on blockchain analysis in this work. In this section, we briefly explain the most common heuristics employed for this.

[18]For example `https://blockchain.info/tags`

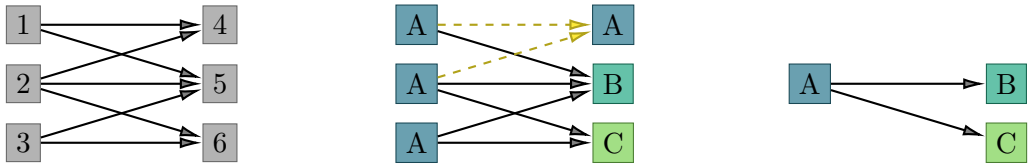


Figure 3.1: **Address clustering:** From addresses to real-world entities: Leftmost graph depicts currency flows (arcs) between six addresses (nodes), a so-called *address graph*. Given that four of the addresses (1-4) belong to the same entity A (determining this is usually the challenging part), it is possible to identify two arcs (from 1 and 2 to 4) as "change" (, ), as seen in the middle. Merging the nodes belonging to the same entity (loops arising from change can be discarded) results in the *entity graph*, depicted on the right.

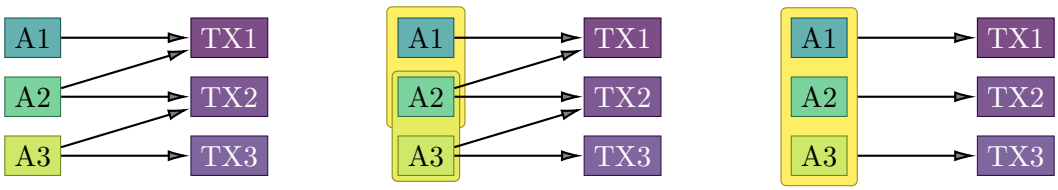


Figure 3.2: **Multiple input heuristic:** If a Bitcoin transaction spends unspent transaction outputs belonging to different addresses (TX1 and TX2), it is usually assumed that the issuer of the transaction is the owner of all of the associated addresses (visualized in the center). The multi-input heuristic then merges address-sets with non-empty intersection, resulting in clusters containing addresses belonging to one entity and its associated transactions.

## 3.1 Multiple Input Heuristic

The *multiple input heuristic* is one of the first heuristics that has been used to analyze the Bitcoin blockchain. The idea has already been mentioned in the Bitcoin whitepaper [Nakamoto, 2008, Section 10: Privacy]. It assumes that a transaction is signed by a single user (or entity), hence the signing user is the owner of all of the transaction inputs, or more explicitly, their associated addresses. This allows to identify transactions in which a user has been involved, even if the user chose different addresses for each of them. This is illustrated in Figure 3.2. The resulting clusters can then be used to derive the entity-graph (see Figure 3.1).

This heuristic may overestimate cluster sizes if the underlying assumption (that it is signed by a single entity) is wrong. This may be the case if a user gives another user access to his assets (giving someone the private key corresponding to the public address instead of sending him the associated assets via a transaction), or for multi-user transactions which may occur via coin mixing or joining (see e.g. [Ruffing et al., 2014, Bissias et al., 2014, Ziegeldorf et al., 2015, Valenta and Rowan, 2015, Maxwell, 2016, Möser and Böhme, 2016]).

**Applicability to Monero:** Monero transaction outputs are addressed to one-time public keys derived from a random value generated during the creation of a transaction and the recipients address. Assuming that neither the sender nor the recipient of a transaction disclose his or her private information[19], the following differences concerning the applicability of this heuristic to Monero apply:

- It is not possible to identify two outputs that belong to the same entity, when they haven't been spent yet.
- Even if it is known that one or multiple addresses belong to the same entity, it is not possible to determine the transaction outputs which belong to the entity.

As each one-time public key only occurs once on the blockchain it is in theory possible to create clusters of these public keys that belong to the same user, though in practice (if the hashing method does not have vulnerabilities) it gives no additional insights.

## 3.2 Change heuristics

As each transaction output is fully spent in a transaction when it is referenced as transaction input, Bitcoin transactions usually have at least two outputs:

1. The desired transfer to the destination address
2. The change sent to a change address, where the remaining coins (those not sent to the destination or used as a fee) from the transaction inputs are sent to

The change address is provided by the creator of the transaction (i.e. the owner of the inputs) to redeem the difference between the sum of the inputs and the desired output (as this difference would else be lost as transaction fee). Several methods to incorporate these change-addresses for address clustering have been proposed.

### 3.2.1 Shadow Heuristic

Using the same change address for multiple transaction would enable linking these transactions as well as all their inputs to the same user. To prevent this, wallets generate a new change address, called "shadow address" for each transaction.

[Androulaki et al., 2013] introduced a heuristic that identifies these "shadow-addresses", usually referred to as "shadow heuristic", which works as follows: If a transaction has two output addresses, $R_n$ and $R_o$, such that $R_n$ is a new address and $R_o$ has already appeared in previous transactions, it's assumed that $R_n$ is a change address and belongs to the owner of the inputs and $R_o$ does not. This heuristic has been refined in [Meiklejohn et al., 2013],

[19] The private information of the sender is the random value used to create the one-time public key, for the recipient it would be his private view-key.

to prevent false positives that would, according to them, collapse the entire graph into large "superclusters". Their approach was to only classify an address as change address, if it occurs only twice on the blockchain (once, when the change is payed out and once when it is redeemed).

### 3.2.2 Optimal Change Heuristic

This heuristic was introduced in [Nick, Jonas David, 2015] and is based on the fact that Bitcoin wallets usually try to minimize the amount of spent outputs (as not doing so would increase the transaction size and fees). The desired amount of the transaction would therefore not be reached, when not all the inputs were part of the transaction. The change of the transactions is therefore smaller than the smallest input, as otherwise the smallest input would have not been included in the transaction.

**Applicability to Monero:** While address clustering does not make sense for Monero (see Section 3.1), identifying change outputs of a transaction could still be useful to model currency flows on the network. To do this, one could proceed as follows: Assuming a transaction $A$ has outputs $A_1$ to $A_m$, some of which are spent in transaction $B$, resulting in outputs $B_1$ to $B_m$. If another transaction now redeems a set $B_i \forall i \in C_I$ together with the remaining outputs from $A$ (or a subset of them), one can assume that (at least) these outputs of $B$ are change outputs. In practice, this method could only be applied if a significant amount of outputs could be linked to the transaction where they are spent. Furthermore, as transaction values are hidden since the v6 hard fork in September 2017 (see Table 2.1), even the currency flows could at most be analyzed in a binary variable. The optimal change heuristic cannot be applied at all since then.

## 3.3 Transaction Fingerprinting

While all (valid) transactions must comply with the format specified by the Bitcoin protocol, there is some leeway (e.g. wallets handle UTXO-choice and change addresses in different ways) which results in some differences which may enable recognizing different transactions from the same user.

**Applicability to Monero:** We are currently not aware of any efforts to use transaction fingerprints for analyzing Monero transactions. Additionally, since the v7 hard fork in April 2018 (see Section 2.2.5), transaction inputs must be ordered by their key image, which prevents leaking any information in the case of transactions using idiosyncratic input-orders[20]

[20] Previously, inputs were ordered according to size, though the roll-out of RingCT transactions prevented this as the amounts were hidden.

CHAPTER 4

# Monero Analytics Techniques

The application of cryptographic methods (see Section 2.2.2) makes analyzing the Monero blockchain more challenging. Due to the unlinkability, it is neither possible to determine which transactions were issued by the same entity, nor is it possible to identify the owner of an output (i.e. the public key where the funds have been sent to). Nevertheless, when two outputs are spent in the same transaction, one could assume that they belonged to the same entity. To prevent this kind of analysis, Monero transactions reference sets of transaction outputs, where only one output is really spent and the others are only referenced as a disguise, so-called mixins. This obscures the transaction graph and is referred to as untraceability.

In 2014, the Monero Research Lab published a note [Noether and Mackenzie, 2014] that outlined possible attack vectors against untraceability, mainly stemming from a malicious user (MU) disclosing his inputs with 0-mixin transactions and thus reducing the size of the anonymity set of inputs in transactions that used inputs from the MU as mixins. Additionally, they considered that this could occur naturally, as users are incentivized by lower fees to choose a small amount of mixins. If users opt to choose 0 mixins for a transaction they deem "uncritical", they not only forfeit on the privacy of their own transaction, but also compromise the privacy of all transactions that uses or used their exposed output as a mixin.

For this purpose, the mandatory minimal ringsize (ringsize = #mixins+1, the "+1" stemming from the real input) has been gradually increased (the most recent increase to a ringsize of 7 has been agreed upon during the developer meeting in March 2018[21] and rolled out to the client with the *Lithium Luna* release[22].

In January 2017 Monero introduced a new type of transaction, called *RingCT*, which also hides the output amount. As RingCT transactions can only use other RingCT

[21] https://monerobase.com/wiki/DevMeeting_2018-03-04

[22] https://github.com/monero-project/monero/releases/tag/v0.12.0.0

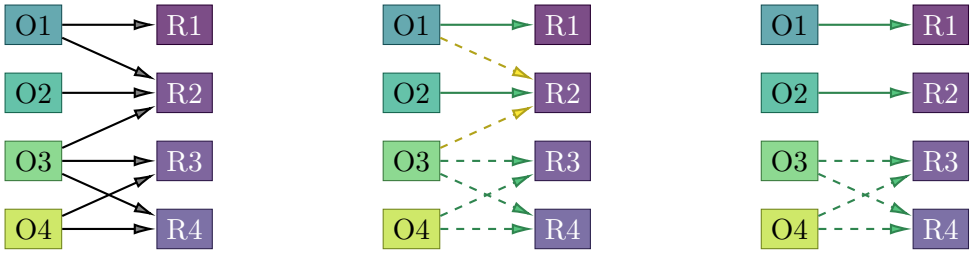


Figure 4.1: **0-Mixin and Intersection Removal:** On the left, a few TXOs (O1-O4) and rings (R1-R4) are depicted, as found on the blockchain. Edges represent references of TXOs in a ring. R1 only references one O1, which must therefore its real input ( ), whereas all other references to O1 must be mixins ( ) and can therefore be removed. Intersection removal can be applied if $n$ rings contain $n$ distinct TXO references. This is the case (for $n = 2$) at R3 & R4 which both reference O3 & O4. O3 and O4 are therefore spent ( ), though it is not known where exactly. In the center, all edges are marked according to their status derived by these methods, on the right the state after the traceability analysis is depicted.

transactions as input, the risk of using already exposed inputs as mixins was reduced as a side effect, as at the time of their introduction (January 2017) there was already a mandatory minimum mixin policy in place (with a minimum ringsize of 3).

In this chapter we present several approaches found in the literature to reduce the ringsize of transactions. To simplify terminology, *input* in the following subsections refers to a reference to a transaction output which may or may not be spent in the transaction. A set of inputs referenced together in a transaction, where one input is real and the others are mixins, is called a *ring*.

If some members of a ring are identified as mixins, we refer to the number of remaining inputs as *effective ringsize*. Obviously, if the effective ringsize of is reduced to 1, the sole remaining input must be the real one.

## 4.1 Iterated 0-Mixin Removal

A Monero transaction with a ringsize of 1 (i.e. 0 mixins) is trivially linked, as the sole input must also be the real one. Matching those inputs to the rings where they are spent with absolute certainty enables the removal of these inputs from other rings, reducing their size by 1 each time. This method is illustrated (together with Intersection Removal) in Figure 4.1. If a ring has only one input left after such a reduction, it is again possible to identify the remaining input as real. This chain reaction has already been considered in [Noether and Mackenzie, 2014], though they most likely underestimated the impact from this. In 2017, two independent studies ([Möser et al., 2018, Kumar et al., 2017]) analyzed all Monero transactions and found that they could identify the real input of in majority of them.

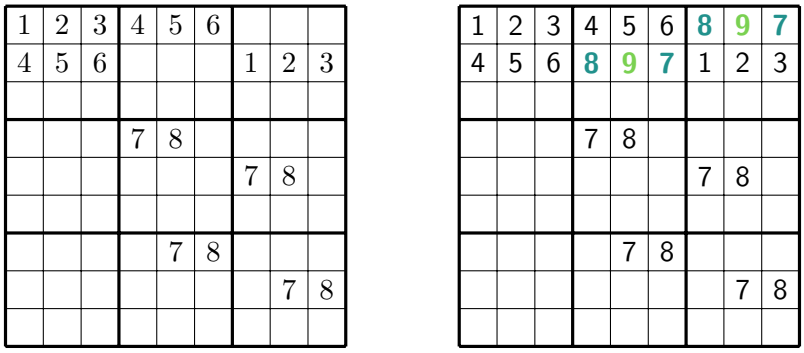


Figure 4.2: **Intersection Removal for Sudokus:** In the Sudoku given on the left, 7,8, and 9 must be in the third row in the upper left box. None of these digits can therefore appear in the other two boxes intersecting the third row. Additionally, using the 7s and 8s in the rest of the Sudoku, one can infer where the **blue** 7s and 8s should go, determining the position of the **green** 9s.

## 4.2 Intersection Removal

The problem of matching transactions to their real inputs from a set of candidate inputs is similar to finding the correct value for a cell in a Sudoku puzzle given its candidate values. The strategy outlined in section 4.1 would be the most straightforward case, where a certain cell $C$ has only one candidate digit, which can then be removed from all cells in the row/column/boxes containing $C$.

The method can be generalized as follows (considering only $n = 1$ one would get the same method as outlined in Section 4.1): If $n$ rings reference the same set $S = \{I_1, \ldots, I_n\}$ of $n$ inputs, each of these inputs has been spent, though it is likely impossible to determine where exactly (assuming that the owners of the outputs do not reveal where they spent them). It is therefore possible to remove all $I_i \in S$ from all other rings as all of them are spent. Figure 4.1 illustrates this method (together with 0-Mixin Removal). This generalized method is essentially the Sudoku method called "intersection removal", where a digit that must occur e.g. in a certain row inside a box, may be removed from the candidate sets of all intersections of that row with other boxes. See Figure 4.2 for an example of this.

In April 2018, [Wijaya et al., 2018] proposed an attack on Monero privacy using a scheme based on a similar idea: An attacker could take $n$ transaction outputs, where $n$ is the current minimum ringsize, to create a transaction with $n$ outputs and $n$ inputs, where each input references the same $n$ outputs. Doing this several times could lead to a large number of transaction outputs that have provably been spent and would reduce privacy, if they were sampled as mixin. They remark that RingCT transactions with their hidden amounts make this kind of attack easier, as all denominations can be attacked at once (though this also increases the pool of legitimate outputs available).

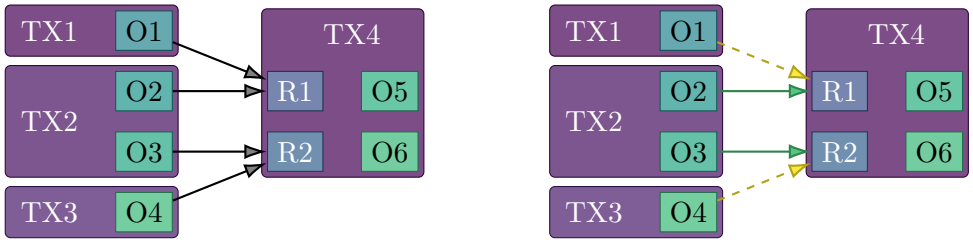


Figure 4.3: **Output Merging Heuristic:** A transaction (TX4) references two outputs (O2,O3) from another transactions (TX2) in two distinct rings (R1,R2). The Output Merging Heuristic then assumes that those are the real inputs ( ). The other ringmembers (O1 and O4) are thus marked as mixins ( ).

## 4.3 Guess Newest Heuristic

In the first years of Monero's existence, mixins have been sampled uniformly from all transaction outputs with the correct denomination. As most transaction outputs are usually spent within a few days after they have been received[23] it is possible to guess which input is real and which are mixins based on their age. [Kumar et al., 2017] and [Möser et al., 2018] both use a very simple heuristic that assumes that for any ring, the most recent input is the real one. To prevent this kind of traceability analysis, the mixin sampling routine has been revised several times, first by sampling from a triangular distribution and later by also enforcing several inputs from the so-called *recent zone*, which has been initially defined as "less than five days old" and (as a reaction to the previously mentioned publications) since September 2017 is constrained to outputs that are "less than 1.8 days old". For more information about all the changes related to mixin sampling please refer to Section 2.2.5.

## 4.4 Output Merging Heuristic

As mixins were chosen from the (usually large) set of eligible transaction outputs, [Kumar et al., 2017] assumed that it is rather unlikely that a transaction has two inputs which reference transaction outputs from the same transaction. They therefore assume that, if this happens, it is because a single entity is the recipient of multiple outputs and this recipient decides to spend both or more of them at the same time (see Figure 4.3). Strategies to avoid these attack vectors have been developed by GitHub user @kenshi84 in Dec. 2016 and accepted in January 2017[24], though only as option (if the "print-ring-members" switch is set to true in the Monero wallet software, users are warned if they try to spend outputs that stem from the same transaction or from multiple transactions from similar blockchain heights).

[23] Using the data from the empirical part of this work, we find that from 2014 to 2017, $\approx 40\%$ of outputs were spent less than 24h after they've been received.

[24] `https://github.com/monero-project/monero/pull/1492`

CHAPTER 5

# Exploiting Hard Forks for Monero Traceability Analysis

If a currency hard fork occurs (see Section 2.1.5), the blockchain splits up into two paths. Users that owned currency before the fork may then choose to redeem their funds (see Section 2.1.5.4) on both chains, either because they decide to use both versions of the currency or because they want to "cash out" on one or both currencies. Irrespective of their motivations, their transactions on the forked chain can reveal information, which, in the case of CryptoNote based currencies such as Monero, may compromise the untraceability guarantee. In this chapter, we will explain how information from two blockchains with a common history can be used to identify mixins in transaction inputs. In the next chapter we will quantify the effectiveness of this method, which we call (due to a severe lack of creativity) *cross-chain analysis*.

## 5.1 Cross-Chain Analysis: Theory

Each time a Monero transaction output $r$ is spent, it is embedded in a ring $R$, together with a (possibly empty) set $M$ of other outputs (the mixins), and a ring signature and a key image are calculated. The ring signature can be used to confirm that the key image is in fact generated from one of the referenced transaction outputs and that it is not only a random character sequence, i.e.:

| | |
|---|---|
| $R = M \cup \{r\}$ | $r$ is the real input and $M$ is a set of mixins |
| $RS = f(R)$ | The ring signature depends on the elements of R |
| key image $= g(r)$ | The key image depends only on the real input |
| $c(RS, g(x)) = \begin{cases} \top & \text{if } x \in R \\ \bot & \text{else.} \end{cases}$ | Confirms that key image belongs to a ring member |

The key image, which is used to prevent double spending as each key image may only occur on the blockchain once, is the key to our approach.

If a hard fork occurs, there are two distinct blockchains, $L$ (legacy) and $F$ (fork), with some shared history, up to block height $s$. The blocks of the $L$ blockchain are called $L_i(i \in I_L)$, the blocks on the forked blockchain are called $F_i(i \in I_F)$. It follows that $F_i = L_i, \forall i \leq s$, i.e. the blocks from the origin up to the split height are identical. If a transaction output from the shared part of the blockchain is not yet redeemed at the time of the split, it may be redeemed on both chains, e.g. in block $F_f$ for $f > s$ and $L_l$ for $l > s$. This is because nodes on either chain only verify that the key image of the output has not been on the part of the blockchain accessible by them (which is either $F_i, \forall i \in I_F$ or $L_i, \forall i \in I_L$).

If a ring $R_F$ on the blockchain $F$ and a ring $R_L$ on the blockchain $L$ have the same key image, the following observations apply:

- The real input must be in the intersection of the two rings[25], i.e. $r \in (R_F \cap R_L)$.
- If $R_F \neq R_L$, elements in $R_F - R_L$ and $R_L - R_F$ (i.e. the symmetric difference $R_F \oplus R_L$) are identified as mixin (this applies to all inputs that reference transaction outputs from transactions in blocks after the split).
- If the intersection only contains one element $(r)$, the real input is identified.

In each case another heuristic may apply, e.g. if a ring has only one member left, the remaining member must be the real one. Additionally, if a real output is identified, all other references (on the same chain) to this output must be mixins and can be marked as such.

Figure 5.1 illustrates how these different heuristics can work together.

## 5.2 Cross-Chain Analysis: Application

Cross-chain analysis as described above results in a possible information gain for each key image that occurs on two separate chains. As there were two forks in close succession, we not only looked for common key images between the main-chain and each fork, but also between the two forks. Combining all this, we applied our cross-chain analysis method as follows:

1. Repeat the following steps until convergence:
    a) Identify all rings with identical key images. For each key image, where this results in a set of rings with at least two elements, set all ring members as mixin that do not appear in all of those rings.

[25] Or a collision occurred, which is rather unlikely. See Footnote 13

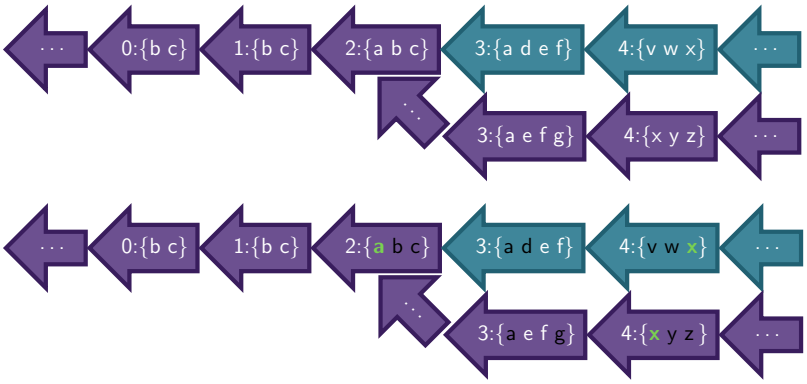


Figure 5.1: **Tracing methods applied in this work:** The upper part of the image is an illustration of the blockchain, two blocks before and the first two blocks after a hard fork. Each block contains one ring, in the format "⟨key image (0-9)⟩{⟨ring members (a-z)⟩}". The first two rings (0,1) have the same two members, i.e. intersection removal can be applied to mark these inputs $(b, c)$ as mixin (black) in ring 2, leaving only input $a$, which is therefore the real (**green**) input. From the two rings with key image 3, input $a$ can be therefore removed as it is spent. Additionally, $d$ and $g$ can also be ruled out as they are not part of the intersection $\{a, d, e, f\} \cap \{a, e, f, g\}$. The intersection of the two rings with key image 4 consists of only one element, $x$, which is therefore the real input.

b) For each chain, separately run Zero Mixin Removal (and possibly other heuristics) until convergence.

c) Propagate information gains from one chain to the other two chains. For this, if an output has been identified as real input for a transaction, set all references to this output in rings with a different key image as *mixin*. Propagation of information between rings with shared key images is covered by a), therefore this is already sufficient.

## 5.3 Multi Chain Intersection Removal

Theoretically, it would be possible to extend the intersection removal algorithm (see Section 4.2) for cross-chain analysis as follows:

1. For each key image (on all blockchains), take the set of all referenced outputs which are not yet deduced as mixin.
2. For each set of referenced non-mixin outputs, count the number of distinct key images
3. If both sets have the same size, we've found an intersection.

In practice, this is very unlikely to happen and, as expected, we did not find any multi-chain intersections in our dataset.

## 5.4 Mitigation Strategies

Starting in February 2018[26], Monero developers implemented several features which enable users to mitigate risks stemming from the analysis method outlined in Section 5.1. These changes were merged and released before the hard forks with client version v0.12 and enabled users to:

- Use the same input sets for transaction on both chains.
- Mix post-split inputs with post-split mixins only.
- Stop using outputs as mixins that are known to be spent.

Ideally, users would redeem their pre-fork funds in a transaction on the Monero blockchain and employ the first feature to create an identical transaction (with the same inputs and ring members) on the forked chain. This prevents to remove ring members that are not part of both rings with a common key image.

The second feature reduces the chance to sample mixins that are revealed due to other users not employing the first feature. To prevent users from suffering a loss of privacy due to the first two measures not being sufficient, the third feature allows users to permanently prevent the wallet software from using certain transaction outputs as mixins (this is referred to as "blackballing").

In Section 6.3 we try to evaluate the adoption of these mitigation strategies for outputs spent on a forked chain. In Section 6.4 we provide three sets of outputs which are either proven to be spent or at risk of being identified as spent and should therefore be blacklisted from the mixin-sampling.

[26] See `https://github.com/monero-project/monero/pull/3322/files` for details.

# Part II

# Empirical Analysis of Monero Traceability

CHAPTER 6

# Monero Traceability with Cross-Chain Analysis

## 6.1 Dataset

To obtain the dataset used in this work we used the most recent release of the Monero daemon[27] to obtain the Monero blockchain from its genesis block up to (and including) block number 1 651 346 (timestamp: 2018-08-31 23:58:15). Additionally, we used the MoneroV daemon[28] to sync the MoneroV blockchain up to block number 1 647 778 (2018-08-31 23:46:43) as well as the Monero Original daemon[29] to sync the Monero Origin/Classic blockchain up to block 1 651 728 (2018-08-31 23:59:45). We then used the transactions-export tool[30] to export the transaction data from all three blockchains and imported it to a PostgresSQL database, where all analysis methods and queries were implemented. We released this toolchain as the MONitERO project on GitHub[31].

Most likely due to a bug in the wallet software, several transaction outputs had public keys that occurred multiple times[32]. For each of these public keys we removed all but the oldest output (according to the timestamp of the block where its transaction is included). All statistics and results in this work are based on this cleaned dataset. In Table 6.1 we list several statistics from our dataset, containing all transactions from Monero, MoneroV and Monero Original up to August 31$^{\text{th}}$, 2018. The values for the two forks are based on those parts of the dataset that are unique on their blockchains[33]. Note that we did

[27] https://github.com/monero-project/monero/releases/tag/v0.12.2.0
[28] https://github.com/monerov/monerov/commit/d3cd9144a1b824aeeb4e2334cf086c962b83f26e
[29] https://github.com/XmanXU/monero-original/releases/tag/v0.11.3.0
[30] https://github.com/moneroexamples/transactions-export/
[31] https://github.com/oerpli/MONitERO
[32] https://github.com/oerpli/MONitERO/blob/master/csv/reused_pubk.csv
[33] Technically, all XMR transactions from block 1 up to the fork-height would also be in the XMO and XMV transaction history.

Table 6.1: **Dataset statistics:** As the Monero (XMR), MoneroV (XMV) and Monero Original (XMO) blockchains share some parts, the values from the two forks (XMV & XMO) only refer to data unique to their blockchain. "Last block" refers to the last block used for the analysis in this work. $\text{XMR}^*_O$ and $\text{XMR}^*_V$ are subsets of the full XMR data set, restricted to the time spans from the XMO and XMV forks.

| | **XMR** | **XMO** | $\mathbf{XMR}^*_O$ | **XMV** | $\mathbf{XMR}^*_V$ |
|---|---|---|---|---|---|
| First TX date | 2014-04-18 | 2018-04-06 | ⇐ | 2018-05-03 | ⇐ |
| Last TX date | 2018-08-31 | 2018-08-31 | ⇐ | 2018-08-31 | ⇐ |
| First block | 1 | 1 546 600 | 1 546 600 | 1 564 966 | 1 564 966 |
| Last block | 1 651 346 | 1 651 728 | 1 651 346 | 1 647 778 | 1 651 346 |
| # Transactions | 4 955 908 | 146 475 | 771 287 | 146 215 | 603 413 |
| # Coinbase TXs | 1 651 347 | 105 729 | 104 747 | 82 814 | 86 381 |
| # TX outputs | 28 878 846 | 198 618 | 1 859 970 | 450 773 | 1 456 810 |
| # Rings (TX inputs) | 24 760 168 | 244 965 | 1 528 763 | 212 919 | 1 182 727 |
| # Nontrivial rings | 12 538 632 | 241 464 | 1 516 342 | 212 919 | 1 175 486 |
| # Ring members | 70 767 723 | 1 243 479 | 11 563 837 | 1 701 036 | 9 014 131 |

include the coinbase TX of the genesis block as its output has been referenced in multiple inputs (4× as mixin and 180× unknown). Rings refer to transaction inputs, consisting of one real input and some ($0 - 4500$ in our dataset) mixins. Nontrivial rings refer to those with at least 1 mixin.

Figure 6.1 shows the monthly number of transactions, for various types of transactions. The rate of coinbase transactions (CB TXs) halves between March and April 2016, due to the doubled block time (since March 22$^{\text{nd}}$, see Section 2.2.5). RingCT transactions overtake regular transactions between Dec. 2016 and February 2017. Virtually all transactions were RingCT half a year before they were made mandatory on September 22nd, 2017. In Figure 6.2 the monthly number of transaction inputs and outputs as well as their per-transaction averages are plotted. In the first few months of Monero's existence there has been a huge number of inputs and outputs. This spike is to a large part due to different denominations (e.g. 912 542 distinct values, in June 2014 alone). After this, the number of inputs and outputs per transaction more or less stabilize (at $\approx 12$ and $\approx 20$), up to the introduction of RingCT. After the introduction of RingCT transaction, the average number of inputs and outputs decrease (as splitting up into denominations is not necessary anymore) to $\approx 2.2$ and $\approx 2.6$, which is in the same range as the values observed on the Bitcoin blockchain.

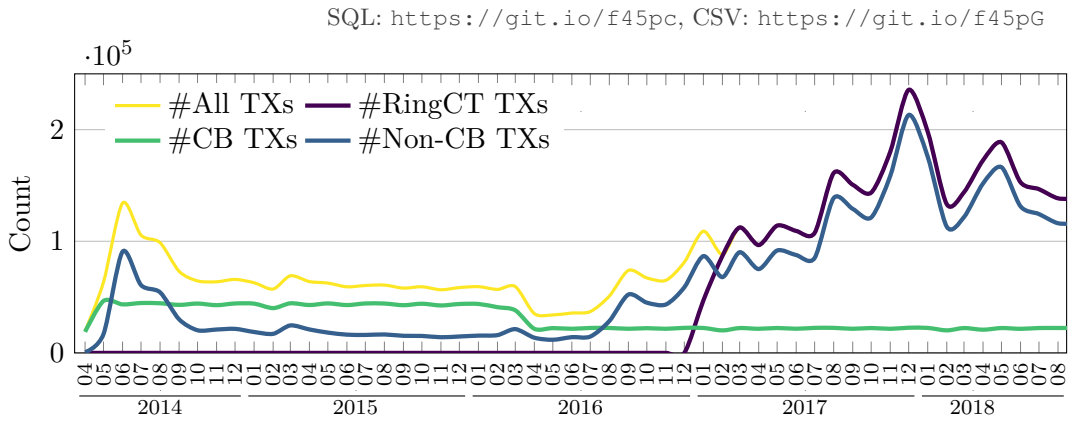


Figure 6.1: **Monero TX-type statistics:** Number of transactions of different types (Coinbase, Non-Coinbase, RingCT and total) issued per month.

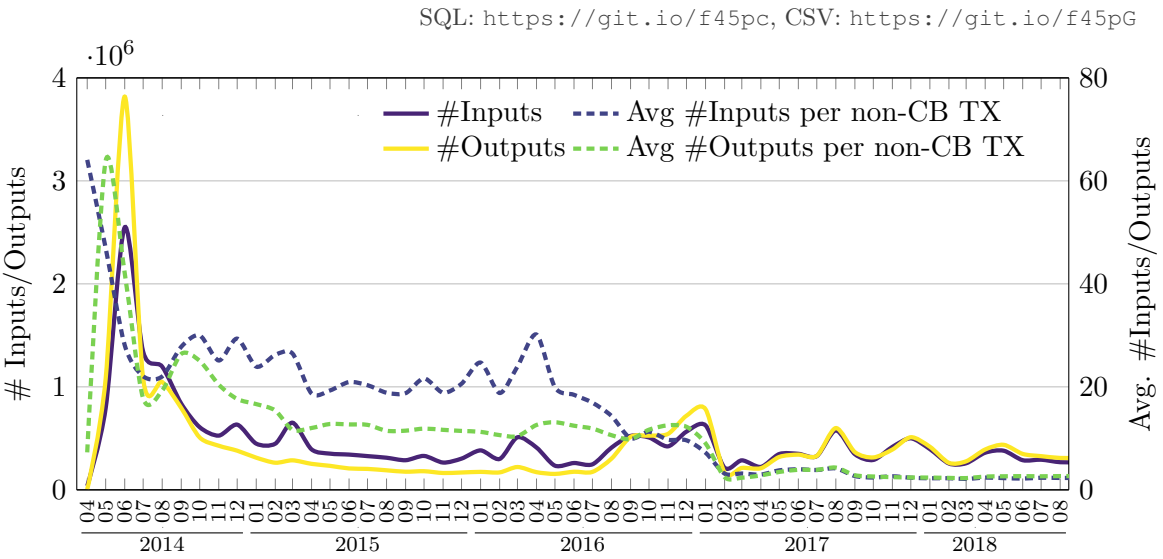


Figure 6.2: **Monero (avg.) Inputs/Outputs statistics:** Number of TX inputs and outputs per month (left y axis, solid lines) and their per-transaction averages (right y axis, dashed lines).

## 6.2 Results

We applied 0-Mixin-Removal (Section 4.1), Intersection Removal (Section 4.2) and the Cross-Chain analysis (Section 5.2) to our dataset to classify ring members as follows:

- *real*: If the ringmember that is spent in an input is identified, it gets marked as *real*
- *spent*: If an intersection set is found, it is (unless the key image of the output is known, e.g. via cross-chain analysis) impossible to know which output is spent in which input, though each output must be spent.
- *mixin*: Decoy ringmembers that are not spent in that transaction.
- *unknown*: If no information is available for this ringmember.

In the following sections, ringsize refers to the number of outputs referenced by an input, as it appears on the blockchain (before removing mixins etc.) and effective ringsize refers to the size of the anonymity set after the traceability analysis (i.e. not counting mixins). If an input has an effective ringsize of 1 (one non-mixin ringmember which would therefore be a *real* ringmember) we refer to that input as *traced*.

Over the whole dataset, we have 70 767 723 Monero ringmembers in 24 760 168 rings, 12 538 632 of which are nontrivial (rings with mixins, i.e. a ringsize over 1). We found 16 433 958 *real*, 16 270 257 *mixin* and 13 240 *spent* ringmembers. These values and the corresponding values for MoneroV and Monero Original can be found in Table 6.2.

Average (effective) ringsize statistics over time are plotted in Figure 6.3. Additionally, the percentage of inputs which use the minimal allowed ringsize[34,35] is plotted ([ ---- ]). Figure 6.4 shows the number of total inputs and nontrivial inputs over time. With the introduction of mandatory mixins these two bars almost converge. The shaded parts of both barcharts show the number of inputs/nontrivial inputs that can be traced. The percentage of traced nontrivial inputs is also plotted ([ ---- ]). Note the small peak in April and May 2018, which results from cross-chain analysis. For the 1 565 858 transaction inputs in 685 608 (non-coinbase) transactions that have been issued since 2018-04-01, this new approach enabled the identification of 73 321 real ringmembers, compared to the 25 256 identified *real* ringmembers without it. The number of identified mixins in this time span has also more than doubled, from 203 251 to 544 131.

In Figure 6.5 the monthly ringsize and effective ringsize distributions from Monero are plotted. Additionally, (as in Figure 6.3) the average ringsize ([ ---- ]) and effective ringsize ([ ---- ]) are plotted on the xy-plane (ringsize, date). Figure. 6.6 is identical, except that inputs with a ringsize of 1 are excluded.

[34] Or a smaller one, if not enough possible mixins of the same denomination are available.
[35] The minimal allowed ringsize depends on the block. Values can be found in Table 2.1.

Table 6.2: **Traceability results:** (RM found in an intersection set are *spent*)

| | **XMR** | **XMO** | **XMR$^*_O$** | **XMV** | **XMR$^*_V$** |
|---|---|---|---|---|---|
| # Nontrivial rings | 12 538 632 | 241 464 | 1 516 342 | 212 919 | 1 175 486 |
| # Ring members (RM) | 70 767 723 | 1 243 479 | 11 563 837 | 1 701 036 | 9 014 131 |
| # Traced nontrivial rings | 4 212 422 | 50 861 | 56 456 | 7 671 | 27 844 |
| # Identified mixin RM | 16 270 257 | 230 128 | 497 570 | 49 035 | 295 107 |
| # Identified real RM | 16 433 958 | 54 362 | 68 877 | 7 671 | 35 085 |
| # Identified spent RM | 13 240 | 0 | 0 | 0 | 0 |

Table 6.3: **XMR Traceability results (TXs between 2018-04-01 to 2018-08-31):**

| | **XMR** |
|---|---|
| # Nontrivial Rings | 1 565 858 |
| # Identified real rm. w/o new method | 25 256 |
| # Identified real rm. with new method | 73 321 |
| # Identified mixin rm. w/o new method | 203 251 |
| # Identified mixin rm. with new method | 544 131 |

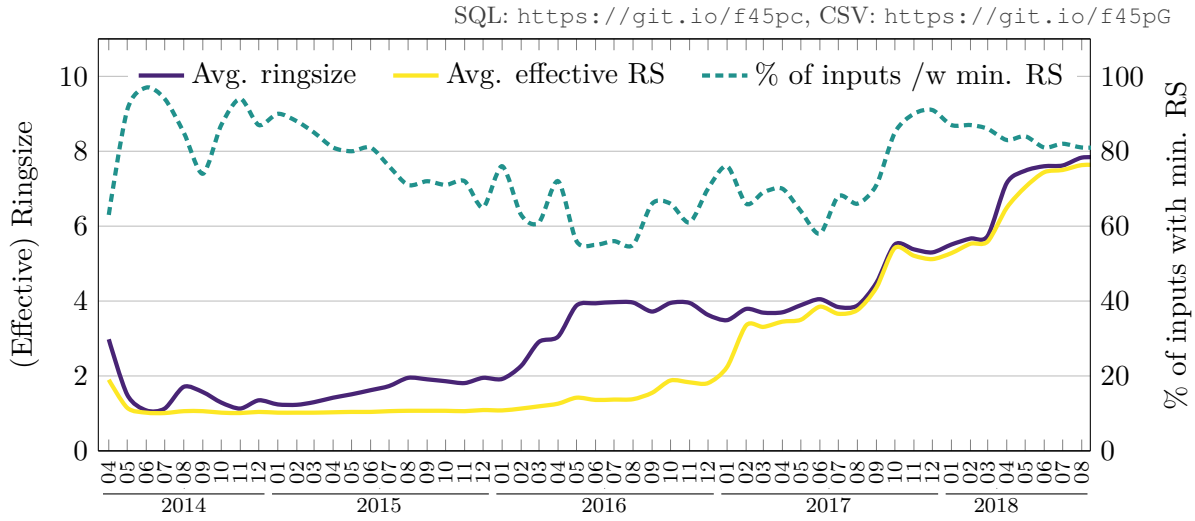


Figure 6.3: **Ringsize statistics:** Average ringsize and effective ringsize of transaction inputs over time (left y axis) and share of inputs (in %) that were created with the mandatory minimum ringsize (right y axis)

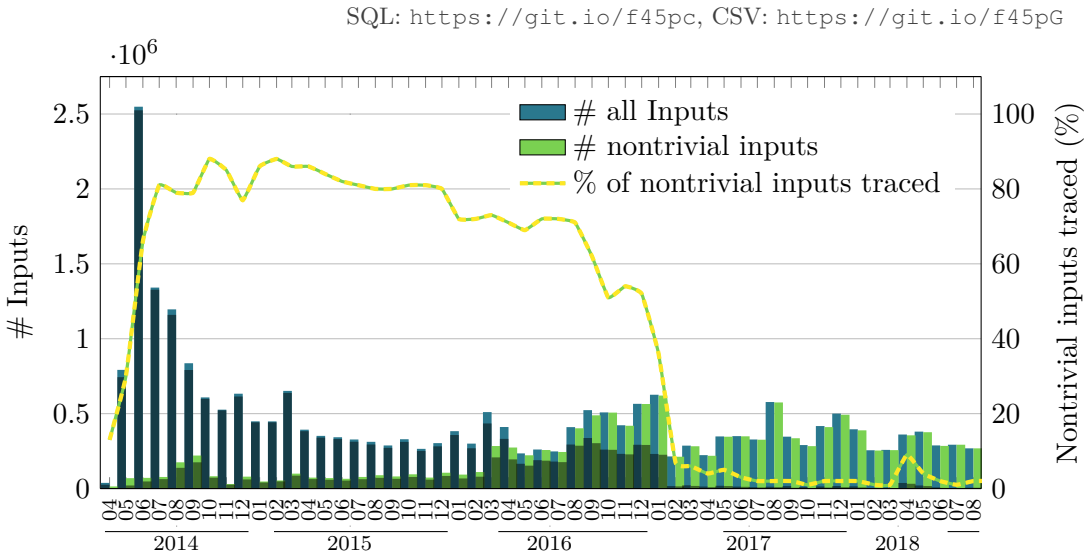


Figure 6.4: **Traceability statistics:** Bar chart of monthly number of all inputs and nontrivial (ringsize $> 1$) inputs (left y axis). In both cases, the traced inputs (inputs where the real spent output is known) are shaded. The percentage of nontrivial inputs which can be traced is plotted as a dashed line (right y axis). In 2014, most of the inputs were in transactions with a ringsize of 1 and were therefore trivially traced. With increasing mandatory minimum ringsizes, the percentage of traceable nontrivial inputs dropped continuously and since the introduction of RingCT, only a tiny amount can be traced. The (small) peak in April 2018 is due to our newly proposed method.

## 6.3 Adoption of Cross-Chain Analysis Mitigation Tools

We try to estimate the adoption of the tools released with Monero v0.12 (see Section 5.4), just before the most recent hard forks, which we analyze in this work. For this purpose we look at the key images of inputs which occur on the Monero blockchain and on one of the forked chains. We find that for XMO there are 56 663 such inputs, whereas for XMV 6 734 matching key images are found. For these inputs we then compare their ring members on both chains and find that 6 497 XMO inputs (11.5%) and 1 231 XMV inputs (18.3%) have matching ring members, which means that in those cases users either correctly applied the tools or used other methods to ensure correct redemption of their funds. This can be seen in Figure 6.7, where a histogram of the ringsizes from the matching inputs for XMO and XMV is plotted. The green part in both cases symbolizes the amount of inputs which were issued with identical ring members on the Monero blockchain as on the fork-blockchain.

Note that for this purpose we are generous with the term *correct*, as we count outputs which have been spent in a ring of e.g. size 5 on the Monero Original blockchain and then with a ringsize of 7 on the Monero blockchain, where 5 of the ring members are identical

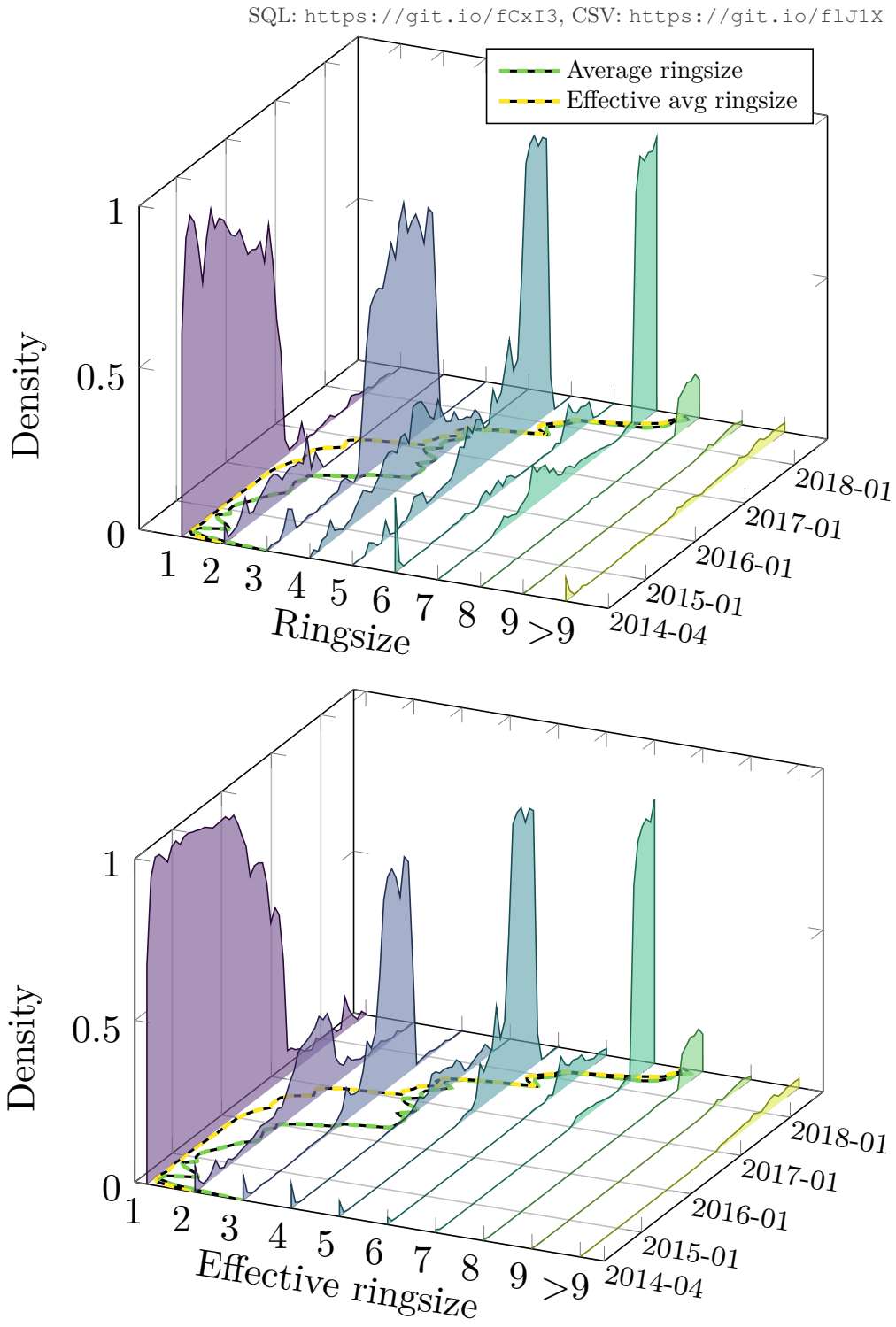


Figure 6.5: **Ringsize distributions over time:** The upper image shows the original ringsize distributions, the lower image the effective ringsizes (after removing identified mixins) . The time measurement (in months) starts at the first block of the Monero blockchain (April 2014). All values are calculated using monthly aggregates. Average from original and effective ringsize basically converge at the introduction of mandatory ringsizes of $\geq 5$.

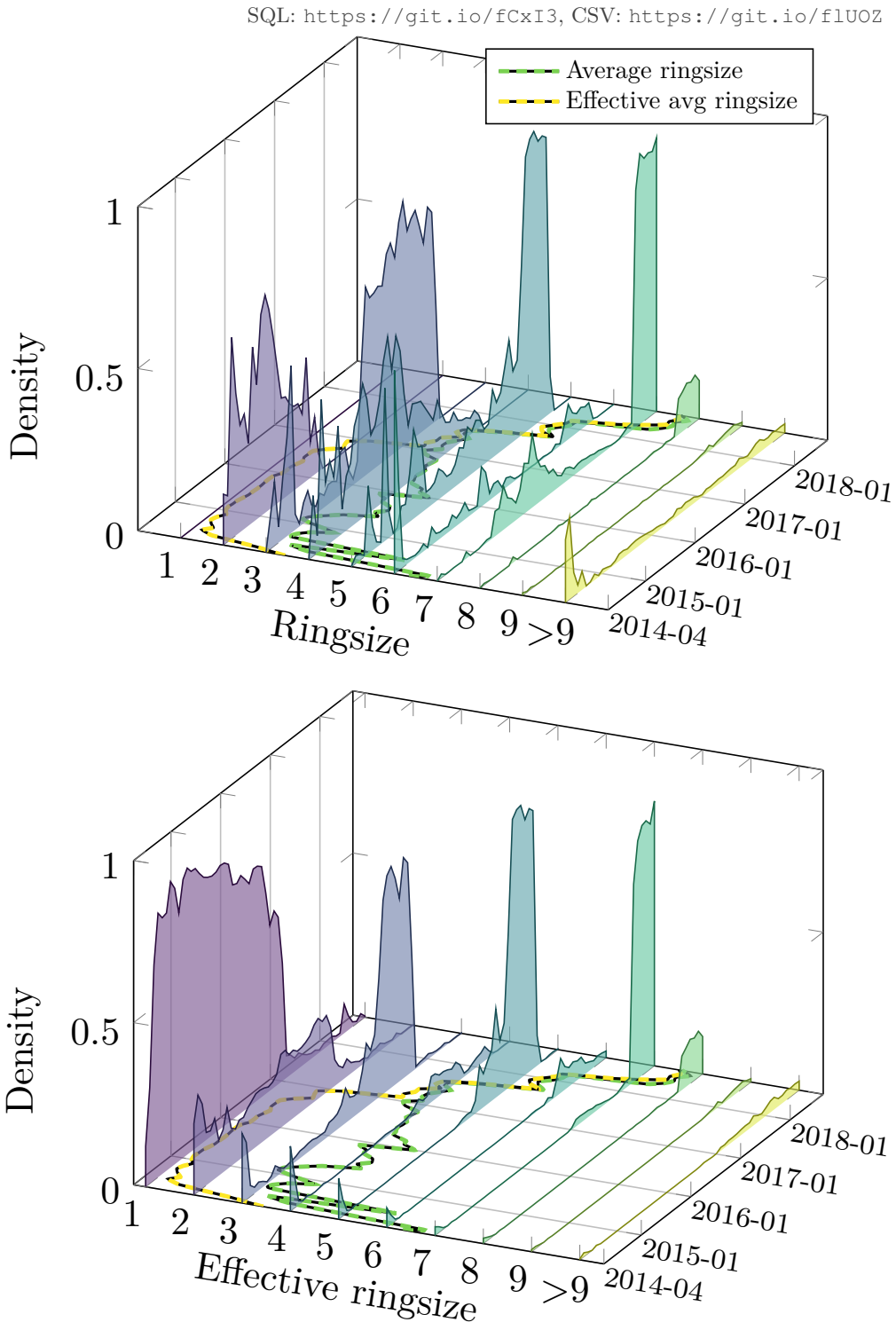


Figure 6.6: **Ringsize distributions over time (excluding trivial inputs):** Same as Figure 6.5 but excluding inputs with a ringsize of 1, which are trivially traced

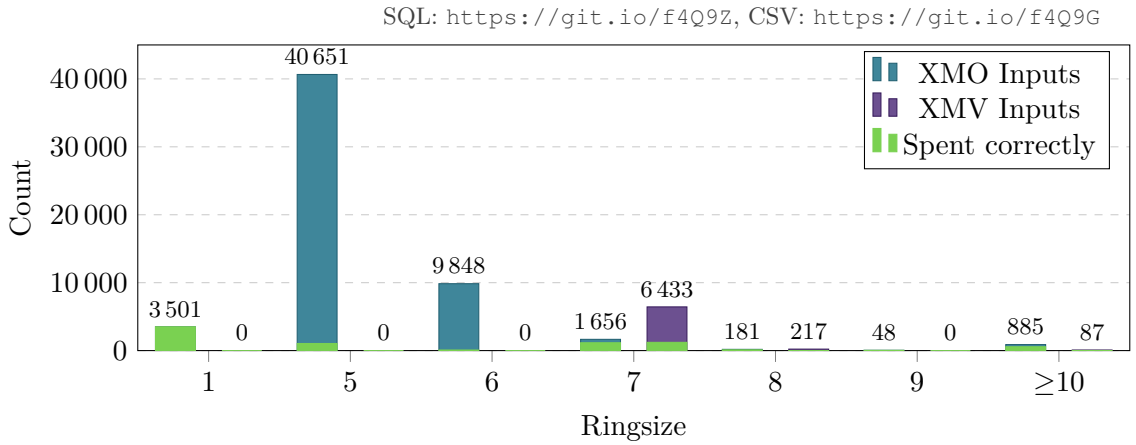


Figure 6.7: **Adoption of CCA-mitigation tools:** Histogram of TX inputs which occur on the Monero and the forked blockchain (matching keyimg, i.e. redeeming the same TX output). The shaded region is the amount of inputs which are issued *correctly* with same mixins on both chains (see definition in Section 6.3). Note the lower minimum ringsize of 5 on the XMO blockchain. Transactions with a ringsize of 1 spend old (pre-RingCT) dust-outputs and are trivially *correct*.

to those on the forked chain and two additional arbitrary mixins. In mathematical terms, if $R_F$ is the ring for a given key image on the forked blockchain and $R_M$ the ring on the Monero blockchain, we define an TX input to be redeemed correctly, if $R_F \cap R_M = R_F$. Our reasoning for this is that users may have accidentally issued a transaction with the old client after the fork date, resulting in a transaction on the XMO blockchain and then chose to make the best out of it by taking the ring from this transaction as basis and add additional mixins to fulfill the higher minimum ringsize on the Monero blockchain.

## 6.4 Spent and Risky Outputs

Since v0.12, the Monero wallet allows users to prevent outputs from being sampled as decoy. The tool has some built-in heuristics that determine some of the public keys that should be avoided. Though, as only the Monero blockchain is considered, outputs that are exposed by being spent incorrectly on a forked chain cannot be identified. For this purpose we release three sets of public keys as CSV files on Zenodo[36]:

- Spent outputs: Outputs that have been provably spent.
- Risky outputs: Outputs that are referenced in an input with an effective ringsize $\in \{2, 3\}$ and are therefore at an elevated risk of being identified as spent.
- Referenced on fork: Outputs that were referenced on either the MoneroV or the Monero Original blockchain and are therefore risky as they could be identified after an additional transaction on the Monero blockchain.

[36] [Hinteregger, 2018a]: `https://zenodo.org/record/1304032`

CHAPTER 7

# Updated Evaluation of Existing Methods

As the Monero development team issued several changes (higher mandatory ringsize and improved mixin sampling) to the transaction protocol to address the concerns raised by the 2017 publications on traceability, we evaluate heuristics applied by [Kumar et al., 2017] and [Möser et al., 2018] using the results from our own traceability analysis.

Additionally, we take another look at some of the results from [Wijaya et al., 2018].

## 7.1 Revisiting the Guess Newest Heuristic

In their analysis, [Kumar et al., 2017] and [Möser et al., 2018] found that most of the time (for the data they analyzed), the real transaction output that is spent in a transaction is the newest one, resulting in the *Guess Newest Heuristic* (GNH) Since then, the mixin sampling has been changed (see Section 2.2.5), so we decided to check whether applying the GNH is still admissible. We looked at all transactions and estimated the accuracy by looking at the status (real, mixin or unknown) of the ringmember that references the most recent transaction output. We found that in 2018, the most recent output is the real input in only 12.1% of the inputs where data is available (13 175 of 109 199). This is similar for transactions after the most recent hard fork (where we identified the real input mostly via cross-chain analysis). Though we think that these results are biased, as the cross-chain analysis technique primarily leads to the identification of rings where pre-fork (i.e. "old") outputs are spent. In Figure 7.1 our results from the analysis of the GNH are plotted. The stacked bar charts show the distribution and total number of newest ringmembers with a given status. If there are multiple (ex-aequo) newest ringmembers, we take all of them into account.

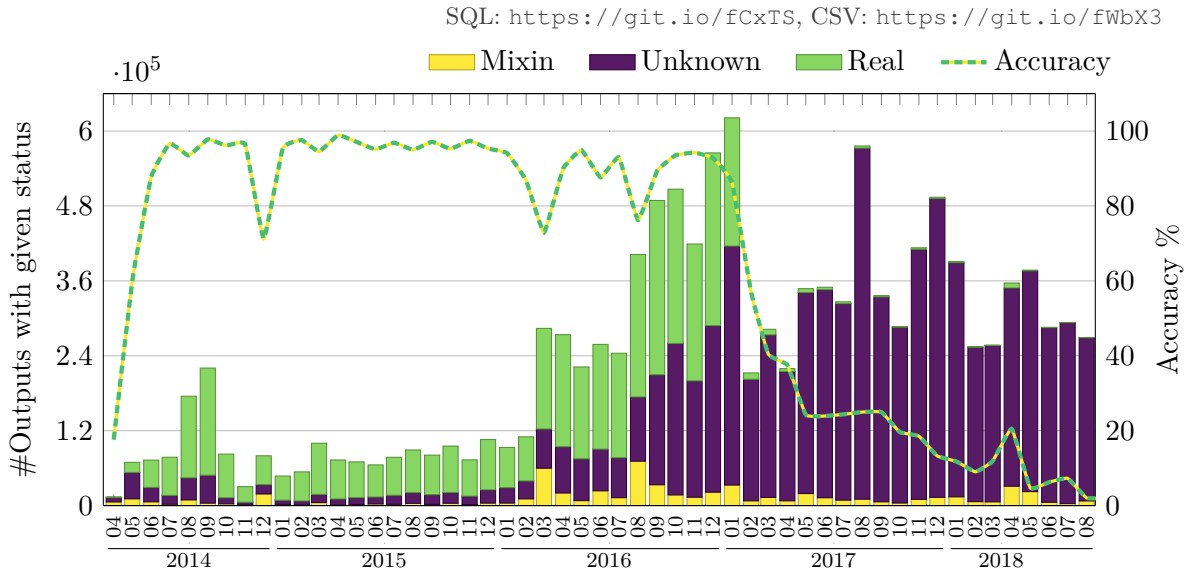


Figure 7.1: **Performance of GNH (see Section 4.3) over time:** After January 2017 the number of identified mixins and real inputs plummet and the accuracy is estimated based on a small sample. As the (crudely estimated) accuracy plummets for recent transactions we do not employ the Guess Newest Heuristic in our analysis.

We believe that the GNH should only be used with caution for transactions after January 2017. While the heuristic may have been admissible up to that date, we did not employ it at all for the results presented in Chapter 6. Our reasoning for this decision was that even at the time where it has been correct in most cases, it could produce false positives which could lead to more false positives for recent transactions, which we wanted to avoid.

## 7.2 Revisiting the Output Merging Heuristic

[Kumar et al., 2017] introduced the Output Merging Heuristic (OMH) which looked at transactions using multiple inputs, where at least two rings reference two distinct outputs from the same transaction.

They found that this heuristic produces correct results in almost all cases, where the ground truth is known (via 0-Mixin Removal), though for transactions with a higher ringsize, the status of the ringmembers identified with the OMH as being real were unknown increasingly often (e.g. for the current minimum ringsize of 7, the split was $\approx 40 : 60$ between true positive and unknown positive).

We applied their heuristic to our dataset and found similar results for older transactions, though for more recent transactions the false positive rate increased. While [Kumar et al., 2017, page 12] found an overall split between true positive/unknown posi-

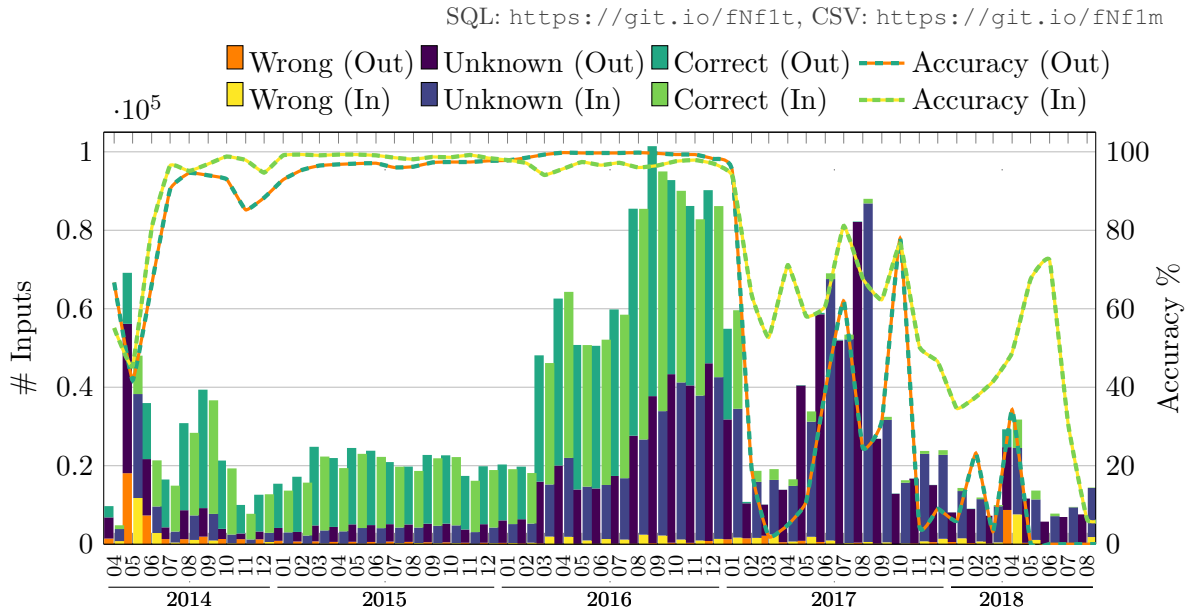


Figure 7.2: **Performance of OMH (see Section 4.4) over time:** Outputs are created at time *out* and spent at time *in*. Left bar for each month uses *out*-time for aggregation, right bar uses *in*.

tive/false positive rate of 87.3/11.9/0.8(%), we find 50.3/46.8/2.8(%) (when not counting inputs without mixins) for the total dataset.

In Figure 7.2 we plot the results from applying the OMH to our dataset. As there are two timestamps whenever the OMH applies (one from the transaction where the outputs are created and another from the transaction where they are spent together), we analyzed both aggregation methods. We calculated the monthly accuracy twice and found rather large fluctuations, though as the accuracy is based on the very small set of ringmembers for which the real status is known, we do not think that this represents more than just random fluctuations. Overall, there do not seem to be any systematic differences between the two aggregation techniques, hence we will only use the timestamp from the combining transaction (where the outputs are spent, labeled "(In)" in Fig. 7.2) for our other figures and measurements related to OMH.

[Kumar et al., 2017] found that with increasing ringsize the share of unknown positives increased while false positives stayed flat (Fig. 11 in their publication). We looked at the status of the ringmembers identified by the OMH as real input for different ringsizes, once for all transactions in the dataset and once for transactions issued since 2018-01-01. The results can be seen in Figure 7.3. We find that on our dataset (blockheight $\leq$ 1 651 346) the heuristic produces more false positives than they found in theirs (blockheight $\leq$ 1 240 503). Also the increase of unknown positives with higher ringsizes vanishes, as this result from [Kumar et al., 2017] is due to their ground-truth being only based on 0-Mixin

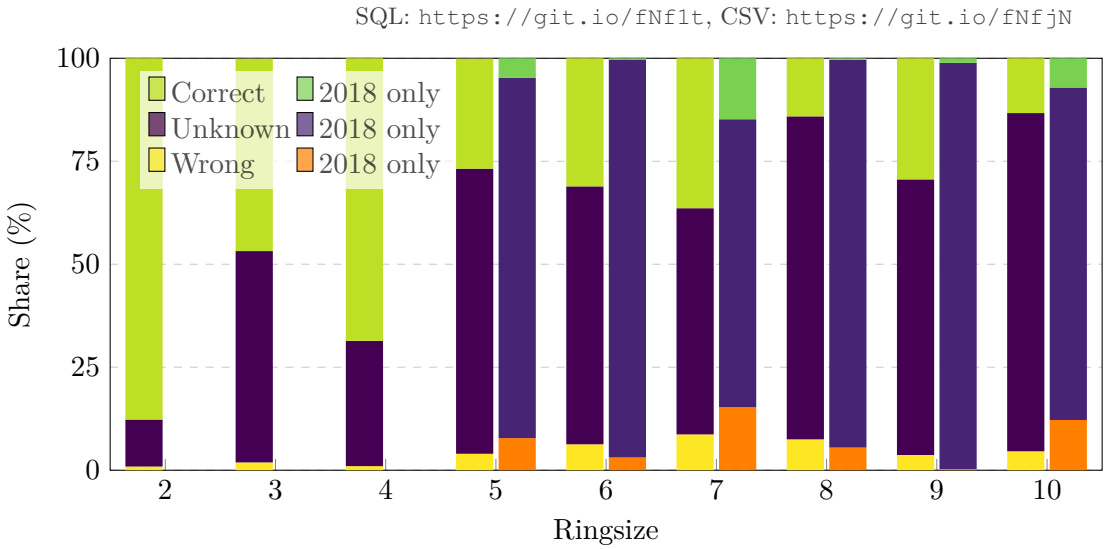


Figure 7.3: **Performance of OMH by ringsize:** Higher rate of misclassification for recent transactions. No obvious relation between true/unknown/false positive rate and ringsize for ringsizes $\geq 5$.

Removal whereas we also used cross-chain analysis, a method that (usually) does not depend on ringsizes.

## 7.3 Intersection Sets

While [Wijaya et al., 2018] only look at rings of size $n$ which occur $n$ times, our approach also finds rings which are identical after some mixins are removed (i.e.: $m$ rings with ringsizes $n_1, \ldots n_m$ and for ring $i$ $n_i - m$ mixins are removed, resulting in $m$ identical rings with an effective ringsize of $m$). We refer to the first kind as *trivial intersections* and to the second kind as *nontrivial intersections*. We compared the intersection sets we found (when restricting our data to the blocks up to height 1 470 000) with those found by [Wijaya et al., 2018] and found similar but not identical numbers. Like them, we find the first trivial intersection in block 47 410 (intersection of size 2, second ring in block 47 416) and the last one in block 1 401 899 (two distinct intersection sets, each of size 5, both transactions in the same block) and the transaction issued by [Wijaya et al., 2018] in block 1 468 439.

Apart from that, we find 1 302 (compared to their 1 244) trivial intersections and 3 005 ring duplicates (2 947) in 901 (885) different transactions.

In Figure 7.4 the number of duplicated rings per month are plotted. The huge peak in September 2014 is from 745 unique intersection sets, each with one mixin, combining dust outputs (in 209 separate transactions). While we do not think that these results

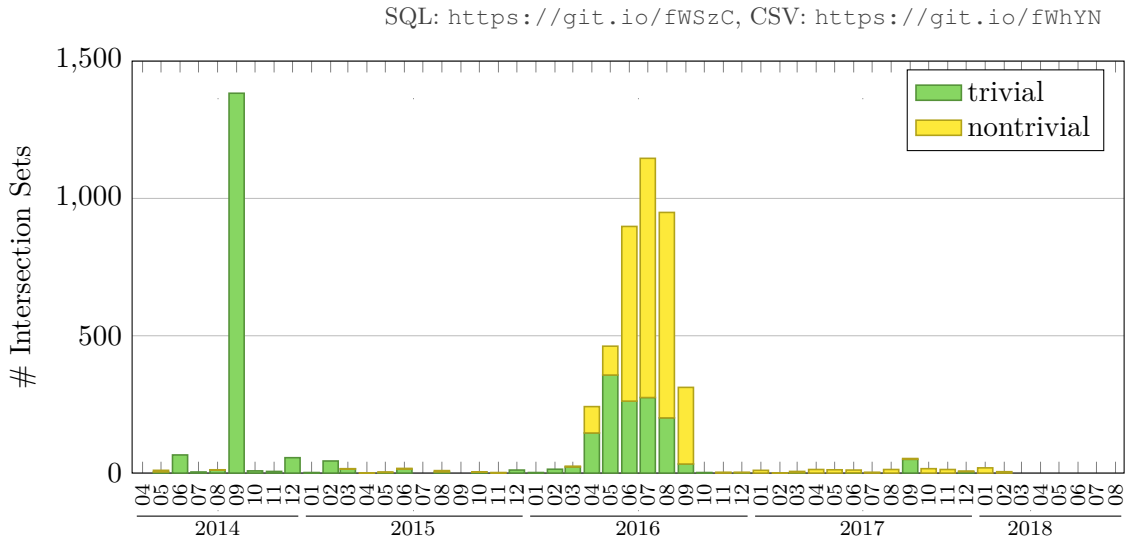


Figure 7.4: **Intersection sets found in dataset:** Monthly aggregates of intersection sets (IS). Trivial IS are identical input rings, whereas nontrivial IS are input rings with a shared subset (the intersection) and some non-shared mixins. Each occurrence of a ring is counted once.

suggest that there has been a targeted effort for a *Monero Ring Attack*, the possibility is there.

Currently the blackball-tools recognize and block outputs referenced in trivial intersection sets ($N$ identical rings of size $N$)[37]. While this approach may recognize some provably spent outputs, it must be noted that attackers could easily circumvent this by creating only slightly more sophisticated intersection-sets. For example, they could take $2N$ outputs $(A_1 \dots A_N, B_1 \dots B_N)$ and create $2N$ different rings of size $N+1$ as follows:

$$\{(A_1, \dots, A_N, B_i) | 1 \leq i \leq N\} \cup \{(B_1, \dots, B_N, A_i) | 1 \leq i \leq N\}$$

We believe that this attack vector is somewhat academic. Nevertheless, the Monero developers and users should be aware that the mitigation strategies implemented so far are not entirely sufficient.

[37] Merged in this PR: https://github.com/monero-project/monero/pull/3428

CHAPTER 8

# Discussion

We analyzed traceability of Monero with the methods proposed by [Kumar et al., 2017] and [Möser et al., 2018] as well as with our own method which exploits currency hard forks. Our new method enabled the identification of the real spent output of 73 321 out of 1 565 858 transaction inputs in the 685 608 (non-coinbase) transactions that have been issued since 2018-04-01 (improved from 25 256). The number of identified mixins in this time span has also more than doubled, from 203 251 to 544 131.

Taken together, the status (real or mixin) of 617 452 out of 11 826 525 ring members in this time frame has been identified, which amounts to 5.22%. (compared to 228 507 and 1.93% without fork analysis). Considering this, we think that there is no huge threat to untraceability stemming from currency hard forks. While it is possible to identify the mixins of a transaction input if the tools provided for securely redeeming airdrops (for creating identical rings on both chains) are not used, the chain reaction from iteratively removing known mixins converges after very few iterations, due to the high enough ringsize.

We therefore conclude that the currently used minimum ringsize of 7 is sufficient to prevent a large chain reaction of identified transaction inputs. Looking at the differences between MoneroV and Monero Original, we find more data that supports this: While the number of transaction inputs (rings) on both fork-blockchains differs by a factor of approximately 1.15 (XMO: 244 965; XMV: 212 919), the number of traced (nontrivial) inputs differs by a factor of almost seven (XMO: 54 362; XMV: 7 671). While this may seem like a case in point for the necessity of a higher ringsize, the data suggests that this is mostly due to the higher amount of incorrectly redeemed pre-fork outputs on the XMO blockchain (50 273 vs 5 506).

We think that redeeming the airdropped funds correctly (with identical rings on both chains) should be sufficient to prevent the identification of mixins. As usage of the cross-chain mitigation tools has been abysmal despite the fact that they have been

advertised heavily on e.g. /r/Monero before the release of MoneroV, we would suggest that the clients should advise users ahead of time on how to employ these tools if there is a fork coming up.

Based on the real outputs which have been identified using the methods mentioned above, we analyzed the performance of existing heuristics. We conclude that temporal analysis in the form of the *guess newest heuristic* should only be applied with caution for recent transactions. While up to 2016 this simple heuristic has been correct in a large majority of cases, the accuracy since then has plummeted and doesn't seem to outperform random guessing for recent transactions (estimated on at least partly traced inputs). This could be due to a biased sample (as most of these inputs were identified via identical key images on the forked chains), the small sample on which the accuracy measurement is based on (as can be seen in Figure 7.1), though we have no hypothesis that would explain this behavior. The performance of the *output merging heuristic* for recent transaction also seems worse than for earlier transactions. The larger problem for this heuristic is the prevalence of RingCT transactions with less inputs and outputs. This leads to less transactions that merge multiple outputs from the same transaction, resulting in fewer possible applications of the heuristic.

# List of Figures

# List of Tables

# Bibliography

[Androulaki et al., 2013] Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., and Capkun, S. (2013). Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer.

[Back, 2002] Back, A. (2002). *Hashcash-a denial of service counter-measure.*

[Biryukov et al., 2014] Biryukov, A., Khovratovich, D., and Pustogarov, I. (2014). Deanonymisation of clients in Bitcoin P2p network. *arXiv:1405.7418 [cs]*, http://arxiv.org/abs/1405.7418. arXiv: 1405.7418.

[Bissias et al., 2014] Bissias, G., Ozisik, A. P., Levine, B. N., and Liberatore, M. (2014). Sybil-resistant mixing for bitcoin. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 149–158. ACM.

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

[Diffie and Hellman, 1976] Diffie, W. and Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654.

[Ermilov et al., 2017] Ermilov, D., Panov, M., and Yanovich, Y. (2017). Automatic Bitcoin Address Clustering. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 461–466. DOI: 10.1109/ICMLA.2017.0-118.

[Eyal and Sirer, 2014] Eyal, I. and Sirer, E. G. (2014). Majority Is Not Enough: Bitcoin Mining Is Vulnerable. In *Financial Cryptography and Data Security*, Lecture Notes in Computer Science, pages 436–454. Springer, Berlin, Heidelberg, ISBN: 978-3-662-45471-8 978-3-662-45472-5, DOI: 10.1007/978-3-662-45472-5_28, https://link.springer.com/chapter/10.1007/978-3-662-45472-5_28.

[Foley et al., 2018] Foley, S., Karlsen, J. R., and Putniņš, T. J. (2018). Sex, Drugs, and Bitcoin: How Much Illegal Activity Is Financed Through Cryptocurrencies? SSRN Scholarly Paper, Social Science Research Network, Rochester, NY, https://papers.ssrn.com/abstract=3102645.

[Fujisaki and Suzuki, 2007] Fujisaki, E. and Suzuki, K. (2007). Traceable Ring Signature. In *Public Key Cryptography – PKC 2007*, Lecture Notes in Computer Science, pages 181–200. Springer, Berlin, Heidelberg, ISBN: 978-3-540-71676-1 978-3-540-71677-8, DOI: 10.1007/978-3-540-71677-8_13, https://link.springer.com/chapter/10.1007/978-3-540-71677-8_13.

[Goldfeder et al., 2017] Goldfeder, S., Kalodner, H., Reisman, D., and Narayanan, A. (2017). When the cookie meets the blockchain: Privacy risks of web payments via cryptocurrencies. *arXiv:1708.04748 [cs]*, http://arxiv.org/abs/1708.04748. arXiv: 1708.04748.

[Gwern Branwen, 2011] Gwern Branwen (2011). Silk Road: Theory & Practice - Gwern.net. https://www.gwern.net/Silk-Road.

[Hankerson et al., 2006] Hankerson, D., Menezes, A. J., and Vanstone, S. (2006). *Guide to elliptic curve cryptography*. Springer Science & Business Media.

[Haslhofer et al., 2016] Haslhofer, B., Karl, R., and Filtz, E. (2016). O Bitcoin Where Art Thou? Insight into Large-Scale Transaction Graphs. In *SEMANTiCS (Posters, Demos, SuCCESS)*.

[Hinteregger, 2018a] Hinteregger, A. (2018a). Monero: Public Keys of spent TXOs. DOI: 10.5281/zenodo.1304033, https://zenodo.org/record/1304033. type: dataset.

[Hinteregger, 2018b] Hinteregger, A. (2018b). oerpli/MONitERO: Version 1.0. DOI: 10.5281/zenodo.1318980, https://zenodo.org/record/1318980.

[Kumar et al., 2017] Kumar, A., Fischer, C., Tople, S., and Saxena, P. (2017). A Traceability Analysis of Monero's Blockchain. In Foley, S. N., Gollmann, D., and Snekkenes, E., editors, *Computer Security – ESORICS 2017*, Lecture Notes in Computer Science, pages 153–173. Springer International Publishing, ISBN: 978-3-319-66399-9.

[Light, 2017] Light, J. (2017). The differences between a hard fork, a soft fork, and a chain split, and what they mean for the.... https://medium.com/@lightcoin/769273f358c9.

[Luigi1111, 2016] Luigi1111 (2016). Understanding Monero Cryptography, Privacy Part 2 – Stealth Addresses. https://steemit.com/monero/@luigi1111/understanding-monero-cryptography-privacy-part-2-stealth-addresses.

[Maxwell, 2016] Maxwell, G. (2016). CoinJoin: Bitcoin privacy for the real world. https://bitcointalk.org/index.php?topic=279249.0.

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

[Möser and Böhme, 2016] Möser, M. and Böhme, R. (2016). Join Me on a Market for Anonymity. In *Workshop on Privacy in the Electronic Society*.

[Möser et al., 2018] Möser, M., Soska, K., Heilman, E., Lee, K., Heffan, H., Srivastava, S., Hogan, K., Hennessey, J., Miller, A., Narayanan, A., and Christin, N. (2018). An Empirical Analysis of Traceability in the Monero Blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3):143–163, DOI: 10.1515/popets-2018-0025, https://content.sciendo.com/view/journals/popets/2018/3/article-p143.xml.

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

[Nick, Jonas David, 2015] Nick, Jonas David (2015). Data-Driven De-Anonymization in Bitcoin. Master's thesis, ETH Zürich.

[Noether and Mackenzie, 2014] Noether, S. and Mackenzie, A. (2014). A note on chain reactions in traceability in Cryptonote 2.0. *Research Bulletin. Monero Research Lab*, (MRL-0001).

[Noether et al., 2016] Noether, S., Mackenzie, A., and Monero Core Team (2016). Ring Confidential Transactions. *Research Bulletin. Monero Research Lab*, (MRL-0005).

[Paquet-Clouston, 2017] Paquet-Clouston, M.-C. (2017). Are Cryptomarkets the Future of Drug Dealing? Assessing the Structure of the Drug Market Hosted on Cryptomarkets.

[Quesnelle, 2017] Quesnelle, J. (2017). On the linkability of Zcash transactions. *arXiv:1712.01210 [cs]*, http://arxiv.org/abs/1712.01210. arXiv: 1712.01210.

[Reid and Harrigan, 2013] Reid, F. and Harrigan, M. (2013). An Analysis of Anonymity in the Bitcoin System. In *Security and Privacy in Social Networks*, pages 197–223. Springer, New York, NY, ISBN: 978-1-4614-4138-0 978-1-4614-4139-7, DOI: 10.1007/978-1-4614-4139-7_10, https://link.springer.com/chapter/10.1007/978-1-4614-4139-7_10.

[Ruffing et al., 2014] Ruffing, T., Moreno-Sanchez, P., and Kate, A. (2014). CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. In *Computer Security - ESORICS 2014*, Lecture Notes in Computer Science, pages 345–364. Springer, Cham, ISBN: 978-3-319-11211-4 978-3-319-11212-1, DOI: 10.1007/978-3-319-11212-1_20, https://link.springer.com/chapter/10.1007/978-3-319-11212-1_20.

[ShenTu and Yu, 2015] ShenTu, Q. and Yu, J. (2015). Research on Anonymization and De-anonymization in the Bitcoin System. *arXiv:1510.07782 [cs]*, http://arxiv.org/abs/1510.07782. arXiv: 1510.07782.

[Valenta and Rowan, 2015] Valenta, L. and Rowan, B. (2015). Blindcoin: Blinded, accountable mixes for bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 112–126. Springer.

[Van Saberhagen, 2013] Van Saberhagen, N. (2013). Cryptonote v 2. 0. https://cryptonote.org/whitepaper.pdf.

[Wijaya et al., 2018] Wijaya, D. A., Liu, J., Steinfeld, R., and Liu, D. (2018). Monero Ring Attack: Recreating Zero Mixin Transaction Effect. Technical report, https://eprint.iacr.org/2018/348.

[Ziegeldorf et al., 2015] Ziegeldorf, J. H., Grossmann, F., Henze, M., Inden, N., and Wehrle, K. (2015). Coinparty: Secure multi-party mixing of bitcoins. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, pages 75–86. ACM.