MONASH University

Anonymity in Cryptocurrency

Dimaz Ankaa Wijaya

A thesis submitted for the degree of Doctor of Philosophy at
Monash University in 2020
Faculty of Information Technology

Copyright Notice

Abstract

A decade after its first launch in 2009, cryptocurrency has become a massive industry with rapid development in its underlying technology, blockchain. Blockchain development is not only focusing on improving its scalability, usability, and robustness, but also on improving users' privacy. Bitcoin, as the first cryptocurrency, proposed a new anonymity scheme to decouple the real users' identities with their Bitcoin addresses. However, this scheme was proven not to provide sufficient privacy; several Bitcoin analyses managed to deduce information about the users. Hence, demand for more anonymous payment system increases.

Privacy-preserving cryptocurrency was introduced to cater for more anonymous financial transaction. This type of cryptocurrency implements several privacy-preserving cryptographic primitives into their protocols. Monero is one of these cryptocurrencies. In Monero, a ring signature becomes one of the core techniques to obfuscate the identity of the sender by adding decoys. A one-time public key (stealth address) mechanism protects the privacy of the receiver by not allowing address reuse.

Although cryptographic primitives are available in the privacy-preserving cryptocurrency, there are remaining identified problems at the protocol level that may affect the anonymity of the users. This research studies Monero and its related protocols to identify and mitigate anonymity problems. We utilise data extraction methods, data analyses, simulations, and experimentation to explore how the cryptocurrency system works and how it impacts the users' anonymity.

We discover that Monero transaction protocol is potent as a tool to attack honest users' anonymity. We also discover that a Monero protocol update that leads to a hard fork exposes its system to anonymity and Denial of Service attack risks. Our findings also indicate that third-party services, such as wallet service providers and mining pools, could become threats to Monero anonymity. Our work proposes potential mitigation strategies on each anonymity threat.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature . :

Print Name . : Dimaz Ankaa Wijaya

Date . : 24 April 2020

Publications

This thesis includes four original papers published in peer reviewed conferences and two paper manuscripts for publication. The core theme of the thesis is anonymity in cryptocurrency. The ideas, development and writing up of all the papers in the thesis were the principal responsibility of myself, the student, working within the Faculty of Information Technology under the supervision of Joseph K. Liu, Ron Steinfeld, and Dongxi Liu.

The inclusion of co-authors reflects the fact that the work came from active collaboration between researchers and acknowledges input into the research.

List of publications included as part of the thesis. The following publications arise from this thesis.

- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu: Monero Ring Attack: Recreating Zero Mixin Transaction Effect. TrustCom/BigDataSE 2018: 1196-1201 (Accepted, CORE Rank A)
- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, Tsz Hon Yuen: Anonymity Reduction Attacks to Monero. Inscrypt 2018: 86-100 (Accepted, CORE Rank B)
- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, Jiangshan Yu: On The Unforkability of Monero. AsiaCCS 2019: 621-632 (Accepted, CORE Rank B)
- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu: Risk of Asynchronous Protocol Update: Attacks to Monero Protocols. ACISP 2019: 307-321 (Accepted, CORE Rank B)

Additional publications. The following articles were published during the course of the project but are not included in this thesis.

- Dimaz Ankaa Wijaya, Joseph K. Liu, Dony Ariadi Suwarsono, Peng Zhang: A New Blockchain-Based Value-Added Tax System. ProvSec 2017: 471-486 (Accepted, CORE Rank B)
- Dimaz Ankaa Wijaya, Dony Ariadi Suwarsono: Cryptaxforensic, When Cryptocurrency, Taxation, and Digital Forensic Collide: An Overview of Indonesian Cryptocurrency Market. CoRR abs/1812.04138 (2018) (Best paper award, NSDF 2018 Call For Paper Competition, Unranked)

- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, Limerlina: Senarai: A Sustainable Public Blockchain-Based Permanent Storage Protocol. CANS 2019: 235-246 (Accepted, CORE Rank B)
- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, Fengkie Junis, Dony Ariadi Suwarsono: Designing Smart Contract for Electronic Document Taxation. CANS 2019: 199-213 (Accepted, CORE Rank B)

Manuscripts. We plan to submit the following manuscripts to future conferences or journals.

- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, Jiangshan Yu: Change in Transactions: A Side Channel Attack to Monero Anonymity. In preparation to submit to IEEE Transactions on Dependable and Secure Computing (IEEE TDSC) journal.
- Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu: Transparency or Anonymity Leak: Monero Mining Pools Data Publication. In preparation to submit to IEEE Transactions on Dependable and Secure Computing (IEEE TDSC) journal.

Acknowledgements

I want to thank my supervisors, Associate Professor Joseph K. Liu, Dr Ron Steinfeld, and Dr Dongxi Liu from Data61 for the guidance, support, and encouragement. My sincere gratitude to my panel members, Dr Li Li, Dr Amin Sakzad, and Dr Shifeng Sun, for the feedback and useful comments to my project. Special thanks to Dr Surya Nepal, Dr Marthie Grobler and Dr Gary Delaney from Data61 for the mentorship and leadership I experienced during my study. I also thank Julie Holden for her academic writing support.

I would also like to thank Monash University and Data61 for providing the best combination of scholarships and an abundant amount of facilities that greatly assisted me during my study.

I acknowledge the ultimate support from my family during my PhD journey that makes the completion of this thesis possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Bitcoin, launched in early 2009, is the first decentralised virtual currency that removes the role of a central authority. A decade after Bitcoin started, the cryptocurrency-related businesses have become a multi-billion-dollar industry. The technology development has progressed so fast that cryptocurrency now has become a new virtual asset class [19]. There are currently more than 2,000 cryptocurrencies available in the market. Bitcoin, the most valuable cryptocurrency, is traded at US$9,667 per coin. The total market value of cryptocurrencies is US$280.7billion[1].

Central authorities run traditional payment systems such as banks. However, cryptocurrency systems employ a different approach to remove the role of a central administration. Instead of keeping the transaction data private, cryptocurrency uses blockchain technology. In a blockchain system, each participant can verify all transactions in the system. Although the transparent system provides benefit to the decentralisation, it also poses new risks to users' privacy. Blockchain technology stores and shares all transaction data to all participants (nodes) connected to the network. Therefore it also makes the transaction data obtainable to the public.

The original Bitcoin blockchain construction describes that there should be no relationship between the real identities of the users and the addresses used in the system [81]. However, several analyses show that Bitcoin transactions may leak users' anonymity [72, 95]. Based on the findings, researchers developed new transaction obfuscation methods in Bitcoin, such as CoinJoin [67] and CoinSwap [68]. Experts also developed new privacy-preserving cryptocurrencies that eliminate privacy problems found in Bitcoin.

[1]Data taken from Coinmarketcap.com on 22 February 2020

Rather than providing add-on privacy-protection features, privacy-preserving cryptocurrencies, such as Monero and Zcash, apply different protocols compared to Bitcoin. Monero exercises linkable ring signature and one-time public key to obfuscate the relationships between the senders and the receivers [108]. Zcash implements zero knowledge-proof protocol to break the relationships between transactions [52].

Although these privacy-preserving cryptocurrencies implemented new advanced algorithms, they still had privacy problems. In Monero, for example, a liquidity problem leads to a devastating impact where more than half transactions are traceable [59, 79]. While in Zcash, users prefer to use a transparent address rather than a shielded address due to computational complexity, thus exposing their privacy [91]. The current system in zero-knowledge proof requires a trusted setup to construct a public parameter [96]. The trusted setup generates a secret key that can create new coins instantaneously without following a proper economic process, which is not favourable to the users [52].

We focus our research on ring signature-based cryptocurrency. In a ring signature-based system, every user can construct a ring spontaneously without any help from other parties [64]. In this research, we take Monero as an example of a ring signature-based cryptocurrency. Monero is one of the major cryptocurrencies in the world with 157 thousand Reddit subscribers[2] and a market value of US$1.2billion[3]. Monero's market cap is far higher than any other ring signature-based cryptocurrencies such as Boolberry and Aeon. Its community constructed a Forum Funding System (FFS) [4] as a means to fund ideas such as software implementation, Monero-related research, and other activities related to Monero. The significant amount of support helps Monero developers swiftly adopt new technologies. This research-focused cryptocurrency is a significant area to investigate, unlike other cryptocurrencies. However, new technology adoptions can introduce anonymity risks to the existing system.

This thesis project aims to improve the users' anonymity when using privacy-preserving cryptocurrency. The improvements are conducted through analyses over the real transaction data, mitigating the problems found, and proposing new protocols to improve the anonymity of the users. We analyse the blockchain data to extract information, formulate the problems and mitigate the problems. We also investigate network communications in Monero systems to discover potential threats to user anonymity.

We examine the usability of the cryptographic primitives in real-world scenarios such as cryptocurrency transactions. The cryptographic primitives might provide security proofs,

[2]Data taken from https://www.reddit.com/r/Monero on 13 May 2019

[3]Data taken from https://coinmarketcap.com/ on 13 May 2019

[4]https://www.getmonero.org/forum-funding-system

but protocol implementations that utilise those cryptographic primitives may develop security challenges and flaws [61]. In Monero, these implementation challenges can also introduce additional sources of protocol-level anonymity leakage. We also study the Monero third-party services, namely wallet services and mining pool services that assist users to participate in the system.

1.1 Privacy and Anonymity

We introduce two key terms we use in the project, namely privacy and anonymity, to understand the context of the project better. Privacy is the ability of individuals, groups, or institutions to control information sharing about themselves to others [88]. Privacy must enable the information owner to withdraw it from public and makes the information anonymous [88]. In order to achieve a state of privacy, anonymity is paramount. The subject anonymity refers to the inability to determine the subject within a set of subjects, which is the anonymity set [88]. The anonymity term is valid in a communication network that consists of sender anonymity set and recipient anonymity set. Any member of the sender anonymity set can send a message to the member of the recipient anonymity set without being identified.

In CryptoNote protocol's whitepaper, van Saberhagen defines anonymity by using two adjacent forms: unlinkability and untraceability [108]. Unlinkability is related to the privacy of the receiver, where different transactions are unidentifiable that they are payable to the same receiver. Untraceability, on the other hand, is related to the sender's privacy. A system is untraceable when it is infeasible to determine the real sender of a transaction.

We also use the term k-anonymity in this project. K-anonymity is a data privacy model where information k within a set of K is indistinguishable among other $k-1$ elements of K [102]. In a ring signature that contains more than one element, the anonymity of each element depends on other elements such that if any elements n is removed from the anonymity set, each remaining element has $k-n$ anonymity. In Monero, k-anonymity identifies the anonymity level of every input containing multiple outputs as the decoys. The anonymity of the real input depends on the indistinguishability of each decoy and the number of decoys used.

1.2 Related Work

Anonymity in cryptocurrency is a subject of interest to many researchers. The pioneering research investigated Bitcoin's anonymity scheme that relies heavily on public-key cryptography and digital signature scheme [72, 92]. Although the users never reveal their identities while using Bitcoin, their transaction patterns and secondary information leak information about their real identities [72, 92]. Privacy-preserving cryptocurrencies, such as Monero, utilise advanced cryptography techniques to prevent transaction analyses. However, research shows that in practice, an information leak can reveal 60-80% of all transactions [59, 73, 79]. The findings indicate that cryptography techniques are insufficient to protect transaction anonymity in real-world use-cases. We will discuss more the background and related work of our project in Chapter 2.

Our project extends the existing works that explore potential protocol-level attacks and threats to transaction anonymity. Our research evaluates the Monero transaction mechanism, the Monero protocol update mechanism, and third-party services. To the best of our knowledge, those research subjects are understudied.

1.3 Problem Statement

The existing research mainly focuses on transaction analyses. We identify research gaps in the area, such as potential active attacks to anonymity, security risks in Monero-specific development style, and anonymity risks by third-party services. We define the research gaps in our problem statement as follows.

`P1`. The users have the freedom to create their transactions in the cryptocurrency system. The users can modify their wallets such that the applications may not follow the best practices to gain the optimum anonymity level. All transaction outputs are candidates for new transaction inputs' decoys or mixins. An attacker can create malicious transactions that produce bad outputs that suffer from anonymity reduction. New inputs that select the attacker's outputs as the decoys may also suffer anonymity reduction.

`P2`. Monero has a semi-annual protocol update, in the form of hard forks. Each time a new software version is released, all nodes must replace their old software to the newest version. However, since the nodes belong to different parties, the protocol update can be asynchronous. The asynchronous protocol update in Monero can result

in anonymity reduction. Potential attacks can exploit the weaknesses caused by the event which target the nodes and the anonymity of the transactions.

P3. In many cases, users rely on third-party services to participate in the Monero system, for example, wallet service providers and mining pools. Third-party service providers may exploit users' data to reveal users' privacy. It is also possible that the third-party service providers deliberately publish private information for their gain.

In order to achieve the aim and solve the problems (P1, P2, and P3) that we have defined, we propose three research questions as follows.

Q1. How can attackers exploit the Monero transaction mechanisms to reduce the anonymity of Monero users?

Q2. What are the risks of the asynchronous protocol update in Monero to the users' anonymity?

Q3. What are the risks of third-party services in Monero to the users' anonymity?

1.4 Contributions

Blockchain technology is gaining in popularity. Industries and governments are finding ways to explore the technology to develop solutions to real-world problems as well as to improve the efficiency of business processes [112, 122, 121]. The security assumption of the blockchain technology is more substantial than the traditional shared database in a multi-stakeholder environment, where multiple verifiers build a more reliable data trust [122]. Governments have also started to explore the possibility of creating their cryptocurrencies to complement the existing fiat currencies [8, 99].

A blockchain system that stores financial information needs a more reliable anonymity guarantee to protect the users' privacy. Ring signature could be one of the possible solutions to provide privacy on the blockchain-based solutions. Blockchain implementation in finance could learn from cryptocurrencies. In terms of providing anonymity, Monero could provide lessons and experiences for future systems. Our research contributes to the research area by analysing anonymity vulnerabilities and potential threats to privacy-preserving cryptocurrency protocols. The details of our contributions are as follows.

C1. We formulate new protocol-level attacks to compromise cryptocurrency users' privacy in the system, namely Monero Ring Attack (MRA) and its extension, Monero Ring Attack Extended (MRAE) [116, 117]. Both attacks exploit Monero transaction creation process, where an attacker removes his or her transaction anonymity to reduce other users' transaction anonymity. We also utilise the Unencrypted Payment ID (UPID) for a traceability analysis and discover more than 300K traceable inputs. The finding is the most significant anonymity problem after RingCT implementation in Monero.

Our analyses identify threats in Monero transaction creation functions that require tighter security measures to provide a more reliable transaction anonymity guarantee.

C2. We formulate issues that occur during Monero protocol update. We investigate three blockchain branches that emerged in 2018 and identify problems of key reuse [119]. We discover more than $50K$ traceable inputs on the main Monero branch. The number of traceable inputs is less significant than our other analyses results. However, we investigate external factors that can influence the scalability of the problems [119], namely market prices and market availability. We propose mitigation strategies to the identified issues by registering blockchain branches and transaction ring members on each blockchain branch [119]. We explore possible Denial of Service (DoS) attack during an asynchronous protocol update[118]. We identify that a non-updating node is prone to a DoS attack, where the attacker floods the node's memory with Monero transactions. The attacker can escalate the DoS attack to target transaction anonymity.

Our analyses identify threats during Monero protocol updates that require data management on multiple blockchain branches.

C3. We explore third-party services in Monero ecosystem, namely wallet service providers and mining pools. We identify risks to transaction anonymity because the nodes can gather sufficient information from the wallets' activities. We also investigate information leak in mining pool-related operations. We utilise lists of won blocks and payouts from Monero mining pools and discover more than 200K traceable inputs. The result is the second-largest anonymity leak after RingCT implementation.

Our study shows a strong need for more dispersed third-party wallet providers. Our results also indicate new requirements for mining pool operators to improve their data accountability without compromising anonymity.

We expect the outcomes of our research can improve the existing systems as well as the future systems. We believe that while blockchain technology provides improvements

in multiple industry sectors, the privacy and anonymity issues in blockchain technology are paramount. Without adequate privacy solutions, blockchain poses a liability to any organisations if it leaks users' privacy.

1.5 Relevance to Cryptocurrency or Blockchain Technology

Our project mainly investigates Monero, which represents a cryptocurrency (blockchain) project that constantly improves its features and efficiency with the following properties.

- Rapid technology adoption and implementation, backed by extensive academic research and financial support from the community.
- A proven and significant cryptocurrency product that has been running for more than six years with a total market value of almost US$1billion.
- Unique characteristics such as scheduled hard forks.

We emphasise the relevance of our project to blockchain technology in general as follows.

- A theoretical cryptographic work must be adopted in a working environment with abundant challenges and obstacles. Adjustments are necessary to deliver desirable goals. A comprehensive approach that considers system-level problem analyses is required. In Monero, ring signature technology relies on a good mixin selection algorithm to produce the best results to the users. In Zcash, founders' reward that was intended to support continuous system development becomes a liability for system anonymity [56].
- Standardised best-practice protocols must be enforced in open-source projects which enable external parties to participate by developing third-party services and applications. Insecure implementations can cause damage to the system, and especially in Monero, can impact other users' transactions. An example of the required standardised protocol in Monero is mixin selection.

1.6 Thesis Organisation

We organise the rest of the thesis as follows. We provide background and related work in Chapter 2 to better understand our research project. We include explanations about blockchain implementation in Bitcoin and its anonymity issues that lead to privacy-preserving technique implementations in cryptocurrency space. We also discuss related work that identifies weaknesses in Monero that helps us to identify the research gaps.

Chapter 3 and 4 address our first research question `Q1` about potential threats in Monero's transaction creation scheme. In Chapter 3, we propose a novel transaction-based active attack called `Monero Ring Attack (MRA)`. `MRA` exploits Monero's transaction creation algorithm to recreate existing attacks called zero-mixin transaction and cascade effect. As a mitigation strategy to prevent `MRA`, we develop an algorithm to search the attacker's transactions and blacklist them.

In Chapter 4, we evaluate the weaknesses of the `MRA` scheme presented in the Chapter 3 and we propose an improved version called `Monero Ring Attack Extended (MRAE)`. `MRAE` is immune to the mitigation strategy that prevents `MRA`. We use the statistical properties of output usage to formulate new metrics called Output Weighting (`OW`) and Input Weighting (`IW`) to mitigate `MRAE`. In this chapter, we also present our `UPID`-based analysis that discovers more than $300K$ traceable inputs. To the best of our knowledge, our `UPID`-based analysis reveals the largest anonymity leak after RingCT implementation.

Chapter 5 and 6 are related to `Q2`, where we investigate risks that follow Monero hard forks, especially two hard forks in 2018. In Chapter 5, we focus on anonymity risks from spending a coin in multiple blockchain branches. Our primary data sources are Monero transactions in multiple branches and market-related information of each cryptocurrency. We then correlate the anonymity risks with factors such as historical market prices and market availability. We discover that financial benefits are the leading cause of anonymity risks of the events.

In Chapter 6, we investigate alternative attacks that exploit Monero hard forks that occur during a protocol update. Through simulations, we prove that Denial of Service (DoS) attacks are feasible to reduce the availability of Monero nodes that runs obsolete software versions. Furthermore, the attacker can escalate the DoS attack into a Monero anonymity attack by transacting the same coins multiple times in different protocols.

The topics in Chapter 7 and 8 address research question `Q3`, where third-party services such as wallet providers and mining pools are of interest. In Chapter 7, we describe how wallet service providers have enough information to de-anonymise users' transactions. Our

review on the available wallet providers corroborates our findings, where alternative wallets such as smartphone-based wallets are prone to anonymity analysis.

In Chapter 8, we address our findings that mining pools leak their users' information and transaction anonymity. Many mining pools publish a variety of information on their websites, such as the blocks they won and transaction payouts to their miners. We extract information from ten mining pool websites and conduct traceability analyses. Through our analyses, we discover more than 200K traceable inputs from our data set. The result is the second-largest anonymity leak in Monero since RingCT implementation. We provide the thesis conclusion and future work in Chapter 9.

Chapter 2

Background and Related Work

This chapter outlines underlying technologies in our research field to provide context for our project. The chapter includes a background of Bitcoin and Monero. We present the privacy issue in Bitcoin, along with proposed solutions to mitigate the problems. We will describe Monero technical details and Monero anonymity issues and problems that initiate our research project.

2.1 Early Work in Anonymous Payment System

In 1983, David Chaum proposed a new cryptographic scheme called blind signature [21] to improve existing electronic banking methods. Chaum predicted that electronic-based services might cause privacy issues for customers [21]. Centralised parties such as banks generally provide an electronic payment method. These parties would have all the information to identify the individual using the payment services, such as locations, associations, and even lifestyle [21]. In his paper, Chaum did not specifically define privacy but described personal privacy as any information about an individual [21].

The blind signature protects the information about the payer and the payee and at the same time proves that the payment is valid [22]. The proposed protocol was applicable not only in the payment system but also in other areas requiring privacy protection such as election systems [21]. The protocol emphasises the untraceability of the sender as the subject conducting the activity [23]. Chaum's work initiated research and development on private electronic payment systems. Cryptocurrencies such as Bitcoin and Monero adopted Chaum's ideas.

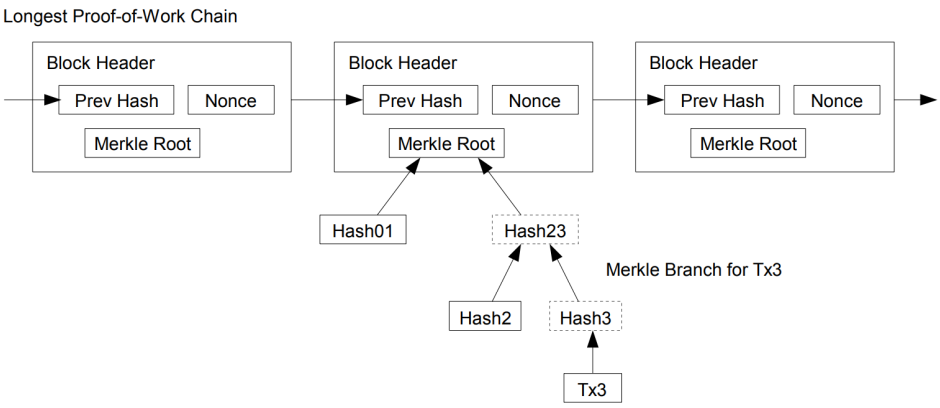

Figure 2.1: A high-level structure of a blockchain [81].

2.2 Bitcoin

Bitcoin is the first cryptocurrency and it surfaced in 2009. Satoshi Nakamoto developed the idea and the code. A year earlier, in 2008, Nakamoto published a whitepaper outlining the construction of the new payment system. This payment system differed from existing payment systems as it does not rely on a central authority. Instead of using a central authority, this new system exercises a consensus method to construct a source of truth [81]. Without any central administration, Bitcoin relies on a peer-to-peer mechanism to maintain the transaction data and update the database whenever new transactions emerge.

Blockchain is a logical construction of chained blocks. The blocks are produced at a specified epochs, depending on the network difficulty, computing power, and luck [3]. A new block connects to its proceeding block by adding a hash value of the previous block into the next block. The basic structure of a blockchain is displayed in Figure 2.1.

2.2.1 Bitcoin Transaction

The transaction scheme in Bitcoin uses an UTXO model (Unspent Transaction Output or unspent output). The UTXO model works identical to how physical coin transaction works. In a transaction, the payer needs to identify which coin to spend, and if the coin value is more than the transacted value, the payer will receive a change. To spend the coins, the owner must provide valid proofs as required by the system. The proofs are the answers to the mathematical puzzle given by the unspent outputs. The evaluation of the mathematical puzzle and the answers looks similar to Forth, a stack-based programming language [3].

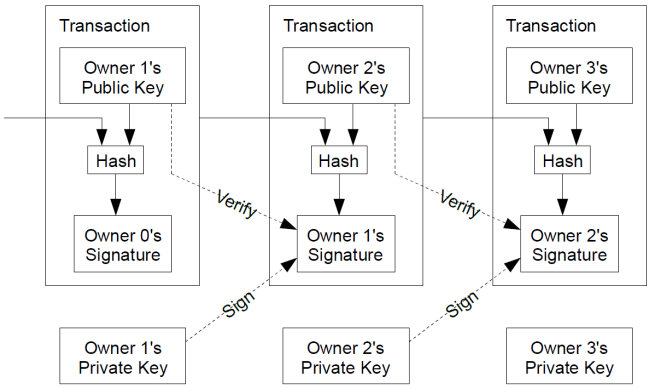

Figure 2.2: Bitcoin transaction visualisation [81].

A user can create a transaction with inputs to spend his or her coins. The transaction will generate new unspent transaction outputs (UTXO). The diagram of the construction is as in Figure 2.2.

When checking a transaction, the Bitcoin system checks the following information:

- Answers to the mathematical puzzle, as stated in the transaction outputs.
- Correctness of the transacted coins. The total number of coins in the outputs should not exceed the total number of coins in the inputs deduced by the transaction fee. The transaction fee is paid to the miners confirming the transaction in the blockchain.

By looking at the Figure 2.2, it can be seen that for each transaction, the coin owner needs to spend all coins in the unspent outputs. Hence, if the total amount of coins to transfer to the payee is less than the total amount of coins in the inputs, the change money is payable to the payer's address. The payer can also create a separate address to hold the change coins.

2.2.2 Bitcoin Privacy Construction

In Bitcoin whitepaper, the privacy model is explained in Figure 2.3. Unlike traditional banking models where all bank accounts are associated with real identities of the customers, in Bitcoin, there is no need to prove that a user has a valid identity. The transactions are validated by using cryptographic methods rather than authenticating the user's identity.

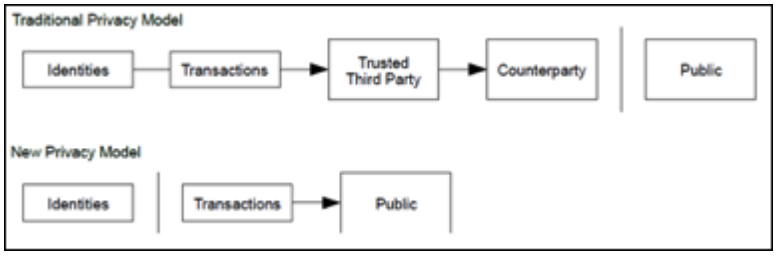

Figure 2.3: Bitcoin transaction [81].

Bitcoin uses Public Key Cryptography, where the hash value of the public key becomes the address, and a valid digital signature determines whether the user is the rightful owner of the fund she claims.

2.2.3 Bitcoin Privacy Problems

In this section, we discuss identified privacy problems and analyses in Bitcoin, namely public key usage, clustering analysis, geolocation, and Know Your Customer (KYC) regulation.

2.2.3.1 Public Key Usage

Research that utilises graph analysis in Bitcoin shows that there are dynamic effects of public keys usage towards anonymity [86]. The public keys can either increase or decrease Bitcoin anonymity. The research focuses on the following items.

- The number of public keys increased since the start of Bitcoin trading in April 2010 and reached its peak in June and July 2011. The research mentions that Bitcoin users start to aware of their transaction anonymity during trading activities that they create new addresses to receive Bitcoins.
- The number of active keys impact the privacy of the users. The active keys are sometimes active only in a certain period, and afterwards, they remain dormant. The dormant coins reduce anonymity because they cannot be in the anonymity set.
- The transaction linkability analysis could produce false results caused by mixing services.

2.2.3.2 Clustering and Off-chain Data Analyses

A group of researchers created an experiment to investigate Bitcoin anonymity level. They created multiple purchases from 26 cryptocurrency exchanges and ten wallet services [72]. They also created payments to 25 different vendors, where nine vendors use BitPay as a payment gateway [72]. The researchers' coins were involved in 344 transactions. From these transactions, they discovered 832 related addresses. From the data set, they developed two clustering heuristics.

- **First clustering heuristic**. If a transaction involves multiple inputs from different addresses, then these addresses belong to the same user.
- **Second clustering heuristic**. The change address in a transaction belongs to the input addresses' owner.

By using both clustering heuristics they developed, they mapped 12 million public keys (16% of all public keys at the time of research) into 3.3 million clusters. The indicates that there is a linkability problem in Bitcoin that correlates different Bitcoin addresses to the same owner.

Separate research combines real-world data with Bitcoin blockchain analysis [39]. The researchers scraped Bitcoin addresses from Bitcointalk[1] web forum by using a Python package called Scrapy and discovered 2,322 unique users that owned 2,404 Bitcoin addresses. The researchers were then able to investigate the activities of identified addresses on the Bitcoin blockchain.

The result shows that the original Bitcoin anonymity model only works in isolation, while external information is useful to uncover the relation between Bitcoin address and the user's identity. Furthermore, in practice, Bitcoin users must share their address to receive coins, either for donation or payment. Although address sharing is unavoidable, it leaks anonymity.

2.2.3.3 Geolocation

Another research project recovered 40% of Bitcoin user profiles. The recovery process succeeded even after the users applied steps to increase their privacy [2]. The researchers evaluated Bitcoin privacy by using two types of heuristic analyses in the first 140,000 blocks of Bitcoin blockchain. The heuristic analyses are as follows.

[1] https://bitcointalk.org

- **Heuristic I Multi-Input Transactions**. The same user owns input addresses of a transaction.
- **Heuristic II Shadow Address**. The shadow address is the address used by a sender to collect the payment change. The sender owns the shadow address.

In addition to the two heuristic analyses, the research also provides an insight into mapping vendors' locations to locate the users in specific regions. The heuristic analyses are useful to increase accuracy. By using Heuristic I, they clustered 1,632,648 addresses into 1,069,699 Generic Addresses (GA). From Heuristic II, they decreased the number into 693,051 GA (58% of Bitcoin addresses). The research also provided ways to evade both heuristic analyses, as follows.

- Evading Heuristic I by the following steps.
  - Minimising coin combination by recreating smaller coins.
  - Combining inputs from several users.
- Evading Heuristic II by:
  - Splitting the coins before sending payments, therefore, change address is not needed.
- Evading geolocation analysis by:
  - Asking vendors to provide a new address for each transaction.

2.2.3.4 Know Your Customer (KYC) Regulation

Know Your Customer (KYC) policy is mandatory in the banking and finance industry. The KYC policy describes that every business entity must keep a record of the identities of their customers before they use the services offered by the business entity. The United States of America was aware that criminals might utilise this type of payment system. Thus they enforced digital currency providers to conduct KYC protocol [80].

In many countries, business entities that provide digital currency exchange services must conduct the KYC protocol. A user who wants to convert fiat money into any cryptocurrencies or vice versa through the platform needs to provide a valid identity document. Therefore, it is easy to link Bitcoin addresses to the owners' real identity [78].

In the United States, cryptocurrency exchanges need to register themselves to the authority and declare their statuses as Money Service Businesses. In this case, they also need to comply with all regulations applied to all traditional financial services [42].

2.2.3.5 Taint Analysis

Bitcoin transactions are traceable due to the transparency of the blockchain data. Everyone can easily trace the activity of a specific address through the blockchain. An observer can identify which addresses sent coins to the monitored address and which addresses received coins from that monitored address. Taint analysis is an analysis over relationships between different addresses [89, 69].

Taint analysis in Bitcoin is possible because Bitcoin implements UTXO model in its transaction. In the UTXO model, the viewer can accurately determine the spent outputs in a transaction. Taint analysis is a common technique to track coin movement in Bitcoin. However, taint analysis could produce inaccurate results if Bitcoin users utilise anonymising protocols such as CoinJoin, CoinSwap, or coin mixers.

2.2.4 Bitcoin Anonymity-Enhancing Protocol

Several protocols to increase the users' anonymity in cryptocurrencies emerged, such as CoinJoin, CoinSwap, the network of transactions, and stealth address. We discuss each protocol in the following subsections.

2.2.4.1 CoinJoin

Maxwell introduced the idea of CoinJoin in 2013 [67]. CoinJoin combines inputs and outputs from multiple transactions into a single transaction. CoinJoin mixes the outputs and the inputs such that the taint analysis is useless against CoinJoin transactions. CoinJoin is available as a separate system to support Bitcoin [66], as well as an additional feature on a privacy-preserving cryptocurrency called Dash [34]. The comparison between standard transactions and a transaction employing CoinJoin is expressed in Figure 2.4.

There are at least two main problems with CoinJoin. First, the number of coins contained in the spent outputs must be equal. In the real world scenario, it would be infeasible to find someone which has the same desire to use CoinJoin and at the same time provides outputs matching the number of coins owned by other users. Second, the security

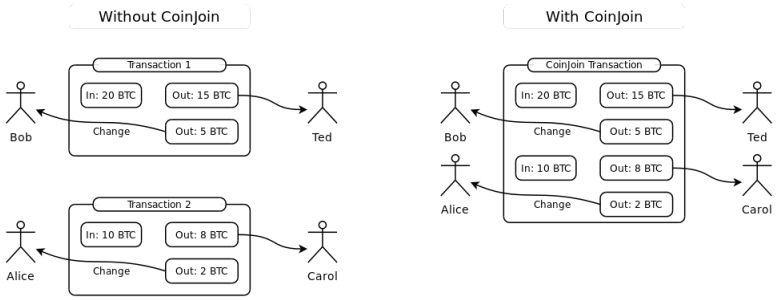

Figure 2.4: CoinJoin scheme [9]

relies on the central authority who knows all the participants and the associated outputs. Hence, there is a single point of failure on the CoinJoin scheme.

2.2.4.2 CoinSwap

CoinSwap is a technique to cleanse tainted coins by swapping the coins with other unrelated coins (Maxwell, 2013b). Maxwell proposed the idea in a Bitcointalk forum post entitled *Transaction graph disjoint trustless trading*. The characteristics of Coinswap are as follows.

- CoinSwap requires at least a middleman to run the protocol where there are two users are exchanging coins.
- The protocol was designed such that no participant could cheat and run away with the coins (trustless).
- CoinSwap is atomic; a participant can cancel the running protocol at any stage without causing loss to other participants.

Coinswap is suitable in the Bitcoin environment. It utilises Bitcoin's advanced scripting language to construct a hash lock contract. All CoinSwap transactions are visible in the blockchain. The construction of CoinSwap is as in Figure 2.5.

A given use case is as follows. Alice wants to send a bitcoin to Bob. However, Alice does not want Bob to know any information about Alice, such as her original address, her current balance, or her source of fund. Therefore, Alice asks Carol to mediate payment to Bob. Alice, Carol, and Bob agree to use CoinSwap protocol. In phase 0, Alice, Carol, and Bob construct escrow transactions that will refund their money if the protocol fails. In

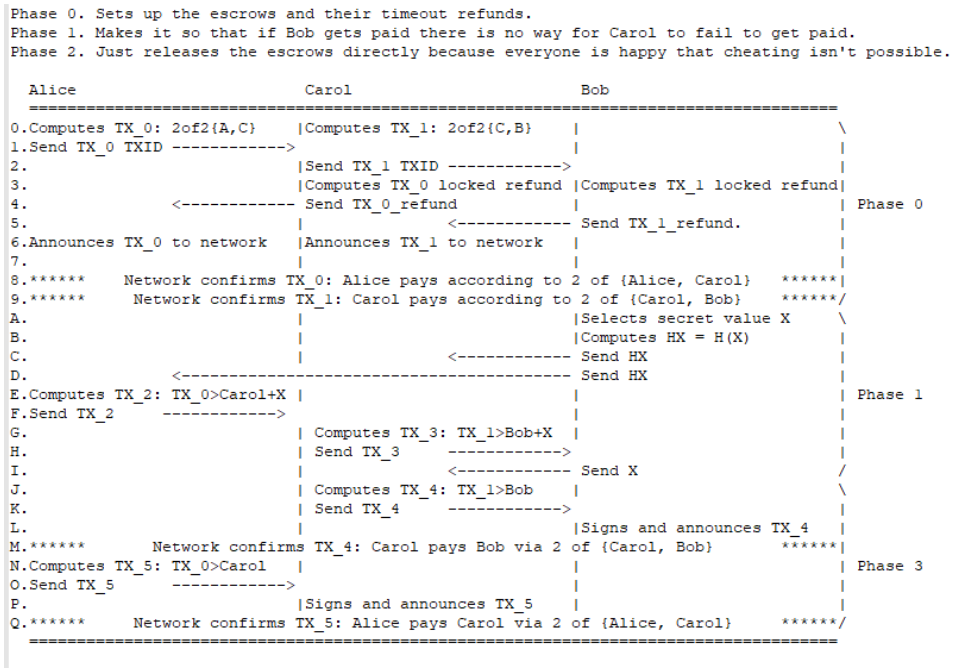

Figure 2.5: CoinSwap Protocol Diagram [68]

Phase 1, Bob generates a random value of X and computes $HX = H(X)$, where H is a hash function. Bob sends HX to Carol and Alice. Alice creates a transaction that pays to Carol if Carol can supply the correct value of X, while Carol creates a similar transaction payable to Bob with the same requirement. In Phase 2, if Bob redeems the payment from Carol, Bob publishes the value of X to the network. Since Carol monitors the network, she now knows the value of X that she can redeem Alice's payment.

The protocol is also applicable in a different way. Alice takes Bob's role to exchange her coin with Carol's coin. Afterwards, Alice pays the exchanged coin formerly owned by Carol to Bob. CoinSwap improves the security of a centrally-operated traditional mixing protocol. A mixing service user relies on the honesty of the mixing operator not to steal the coins that the user deposits to the operator. If the operator cheats and embezzles the coin, the user cannot retrieve the payment, since the transaction is permanent.

CoinSwap has a problem. Carol becomes a participant that knows that such operation has occurred between Alice and Bob. Carol also knows Alice's and Bob's addresses. She can leak the information that relates Alice and Bob by using information she knows for financial benefits or under subpoena.

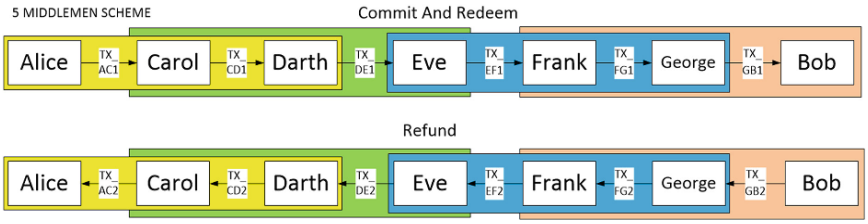

Figure 2.6: Network of transactions with five middlemen. [120]

A similar idea called TumbleBit brings the swapping mechanism out of the blockchain [45]. In TumbleBit, the participants need to create and solve a mathematical puzzle to authenticate the ownership of the tumbled coins.

2.2.4.3 Network of Transactions

The network of transactions enhances the solution on CoinSwap and CoinJoin by replacing a single intermediary into several intermediaries. The purpose of having multiple middlemen is to avoid the middlemen to learn about the transactions conducted by the participants. In the solution, no single intermediary has the full information on how the money is transferred from the sender to the receiver, even if there are at most two collaborating intermediaries to identify the sender and receiver [120]. The scenario is described in Figure 2.6.

There will be multiple transactions confirmed in the blockchain. Hence, it will take more time and cost for the sender to obfuscate the source of money payable to the receiver. Although the protocol ensures that if the protocol halts before its end, all participants will get their coins back.

2.2.4.4 Stealth Address

Stealth address is a method to ensure that the users always use new addresses for every transaction created in the blockchain. In the stealth address technique, the sender creates the receiver's addresses. The receiver sends a parent key, and then the sender will be able to create new addresses based on the parent key and a secret parameter. Later, the receiver needs to scan the blockchain to identify the payment from the sender by using a specific technique.

The idea emerged in 2014 [103] and Dark Wallet became the first implementation of stealth address technique in Bitcoin environment [104]. The protocol utilises Diffie-Hellman

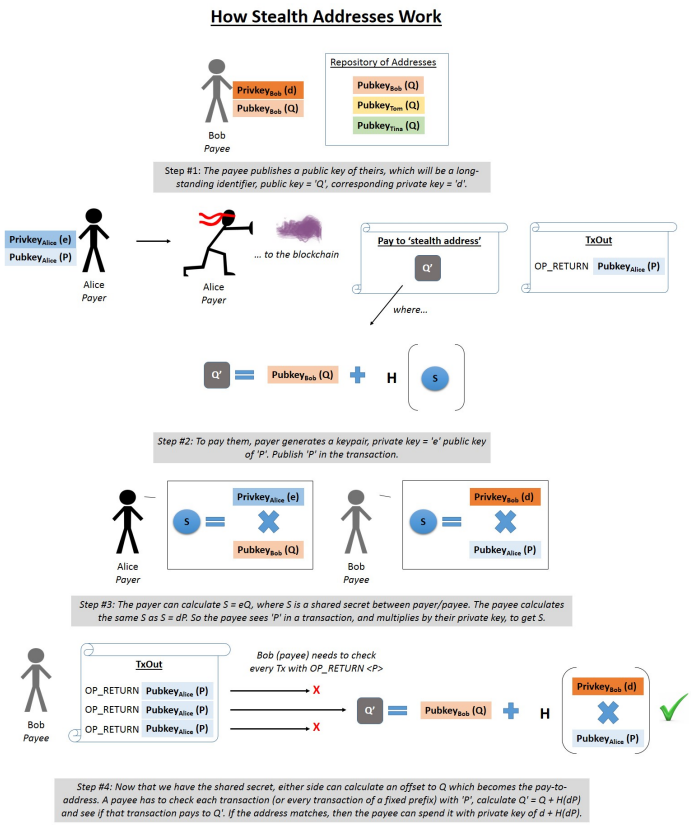

Figure 2.7: Stealth address implementation in Darkwallet [104]

key exchange mechanism to make sure that the receiver can claim the coins sent by the sender. The Dark Wallet implements the stealth address concept in Bitcoin. The Bitcoin's `OP_RETURN` opcode contains the shared parameter of the stealth address. Figure 2.7 shows the stealth address scheme in Bitcoin.

2.3 CryptoNote

The CryptoNote protocol was proposed in 2013 [108]. The purpose of the CryptoNote protocol is to create a privacy-preserving cryptocurrency with built-in features that will help users to keep anonymous. However, it still preserves similarities with Bitcoin, such as transparent transaction data (inputs, outputs, and the number of coins transacted). The

main idea of the protocol is to employ a linkable ring signature [40, 64] to avoid the sender from being traced. The one-time public key (stealth address) guarantees that all users create a new address for every transaction.

By using a linkable ring signature, it is infeasible to distinguish the real output being spent by a transaction over a set of outputs. In the linkable ring signature, the user needs to construct a group of outputs as an input of the transaction, which is assumed to be selected randomly over a broad set of outputs freely available on the network. The user needs to insert her output to the set, which is the real output to be spent in the transaction. The other outputs are the decoys that help to obfuscate the real output. The linkable ring signature also protects the system from a double-spending attack by allowing detection if such an attack occurs. Each public key is associated with a secret value. Its corresponding secret value has to appear in the blockchain if the public key is an input to a transaction. If a secret key appears more than once, it means a double-spending attempt occurs [64].

In the one-time public key mechanism, the receiver provides a set of master public keys to the sender, which will be used by the sender to create new public keys. Hence, the sender creates the destination of the payment, not the receiver. When receiving payments, the receiver scans the blockchain and applies a detection algorithm to determine which payments belong to the receiver.

CryptoNote utilises elliptic curve edDSA [108] introduced by Bernstein [10], with the following parameters.

- $q = 2^{255} - 19$, a prime number.
- $d = -121665/121666$, an element of $\mathbb{F}_q$.
- $E, -x^2 + y^2 = 1 + dx^2y^2$, an elliptic curve equation.
- $G = (x, -4/5)$, a base point.
- $l = 2^{252} + 27742317777372353535851937790883648493$, prime order of the base point.
- $\mathcal{H}_s := \{0,1\}^* \rightarrow \mathbb{F}_q$, a cryptographic hash function.
- $\mathcal{H}_p := E(\mathbb{F}_q) \rightarrow E(\mathbb{F}_q)$, a deterministic hash function.

A private elliptic curve key (private ec key) is defined as $a \in [1, l-1]$, while the public key of a is $A = aG$. CryptoNote uses two public key pair, namely (a, b) associated with (A, B), where $B = bG$. A special key named tracking key is defined as (a, B).

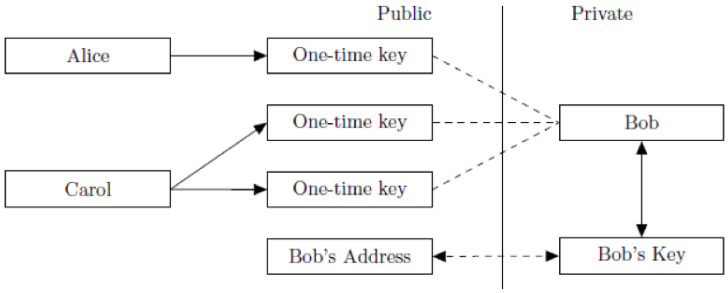

Figure 2.8: One-time key in Monero [108]

2.3.1 One-Time Public Key

The one-time public key employed in the CryptoNote protocol is somewhat similar to the stealth address in Bitcoin ecosystem [103]. In the scenario, the receiver sends a parent public key to the sender. The sender then generates new child public keys by using secret keys. The encrypted secret keys are in the transaction data [84].

The receiver scans the network for new transactions and computes the secret key of each transaction with the parent private key she holds. If the result matches the destination key, then she includes the transaction in her wallet as an incoming transaction. Figure 2.8 shows the mechanism of the one-time key in Monero.

By using the one-time public key, only the sender and the receiver knows the relations between the parent public key and the child public keys involved in the transaction. At the same time, an observer that has access to the blockchain cannot analyse the relationship between the public key and the child keys without any additional data.

2.3.2 Linkable Ring Signature

Rivest, Shamir, and Tauman were the first to propose a ring signature to leak a secret to public [94]. The technique guarantees that the leak originates from a reputable source (e.g. from a company's board of director). However, the person leaking the secret does not want anyone to learn that he or she is the individual leaking the secret. The ring signature enables signing information by using a private key that corresponds to a public key in a set of public keys. Nobody will be able to determine which public key is the one signing the transaction.

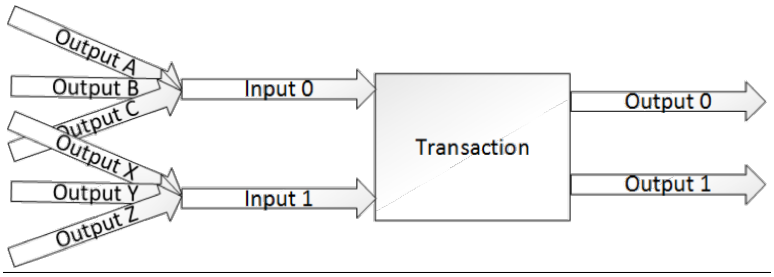

Figure 2.9: The structure of a Monero transaction. Each input is a ring signature that consists of multiple existing outputs (from previous transactions recorded in the blockchain.)

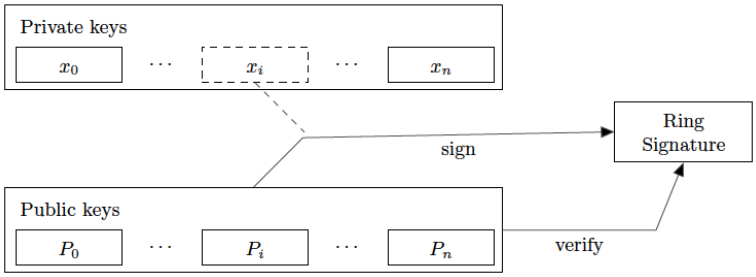

Figure 2.10: The ring signature in Cryptonote [108].

The ring signature construction in CryptoNote originates from previous works on linkable ring signature [64, 63] and traceable ring signature [40]. The constructions ensure that signatures are linkable if they use the same public key. The characteristic is vital in cryptocurrency to avoid double-spending. Double spending is an event where a coin (or a balance) is spent more than once during its lifetime. If double spending can occur in any cryptocurrency, then the coins in the system are worth nothing and cannot be used as a medium to store value [107]. Linkable ring signature limits the anonymity in a ring signature. However, this limitation is useful in many scenarios such as e-voting [24, 124] and cryptocurrency system [101], where a double-vote or double-spending of a coin is undesirable.

In CryptoNote, the ring signature combines several existing outputs (called decoys or mixins) which have the same amount of coins into a single input. These outputs must have a real output in the transaction. A transaction might have multiple inputs and multiple outputs, as well. The high-level structure of a Monero transaction is as in Figure 2.9, while the design of the ring signature is shown in Figure 2.10.

The purpose of the ring signature is to reduce the possibility of an adversary to guess

a real output over several outputs N. We denote the probability (P) of guessing the real output in the input as $P = \frac{1}{N}$.

The signature scheme consists of four algorithms, namely GEN, SIG, VER, and LNK.

GEN. The signer chooses a random secret key $x \in [1, l-1]$. The corresponding public key $P = xG$ and a key image $I = x\mathcal{H}_p(P)$.

SIG. The signer generates the one-time ring signature. A random public keys from other users S' of n, own key pair (x, P), random $\{q_i | i = 0...n\}$, random $\{w_i | i = 0...n\}$ from $(1...l)$. Then, the following transformation based on [28] is applied.

$$L_i = \begin{cases} q_i G & \text{if } i = s \\ q_i G + w_i P_i & \text{if } i \neq s \end{cases}$$

$$R_i = \begin{cases} q_i \mathcal{H}_p(P_i) & \text{if } i = s \\ q_i \mathcal{H}_p(P_i) + w_i I & \text{if } i \neq s \end{cases}$$

A non-interactive challenge is defined as

$c = \mathcal{H}_s(m, L_1, ..., L_n, R_1, ..., R_n)$

The response is computed as

$$c_i = \begin{cases} w_i & \text{if } i \neq s \\ c - \sum_{i=0}^{n} c_i \bmod l & \text{if } i = s \end{cases}$$

$$r_i = \begin{cases} q_i & \text{if } i \neq s \\ q_s - c_s x \bmod l & \text{if } i = s \end{cases}$$

As the result, ring signature $\sigma = (I, c_1, ..., c_n, r_1, ..., r_n)$.

VER. A verifier checks the signature by using inverse transformation as follows.

$$\begin{cases} L'_i = r_i G + c_i P_i \\ R'_i = r_i \mathcal{H}_p(P_i) + c_i I \end{cases}$$

Then, the verifier checks the following equation.

```
Procedure generate_signature(M, A[1], A[2], ..., A[n], i, a[i]):
   I <- a[i]*H(A[i])
   c[j], r[j] [j=1..n, j!=i] <- random
   k <- random
   For j <- 1..n, j!=i
      X[j] <- c[j]*A[j]+r[j]*G
      Y[j] <- c[j]*I+r[j]*H(A[j])
   End For
   X[i] <- k*G
   Y[i] <- k*H(A[i])
   c[i] <- H(H(M) || X[1] || Y[1] || X[2] || Y[2] || ... || X[n] ||
    Y[n])-Sum[j=1..n, j!=i](c[j])
   r[i] <- k-a[i]*c[i]
   Return (I, c[1] || r[1] || c[2] || r[2] || ... || c[n] || r[n])
End Procedure
```

Figure 2.11: Ring signature generation in CryptoNote Standard `CNS002` [109].

$$\sum_{i=0}^{n} c_i \stackrel{?}{=} \mathcal{H}_s(m, L'_0, ..., L'_n, R'_0, ..., R'_n \bmod l.$$

LNK. The verifier checks if I has appeared before. A signature that has the same secret key x_i produces the same I_i.

2.3.3 CryptoNote Standards

CryptoNote Foundation currently maintains CryptoNote protocol. The foundation has developed a set of documented standards[2] called CryptoNote Standard (CNS). We discuss the standards in the following subsections.

2.3.3.1 CryptoNote Signatures

CNS002 [109] describes CryptoNote signature mechanism. CNS001 replaces the obsolete CNS001. There are two main algorithms in CNS002, namely generate_signature and verify_signature. The algorithms are shown in Figure 2.11 and Figure 2.12.

[2]https://cryptonote.org/standards/

```
Procedure verify_signature(M, A[1], A[2], ..., A[n], I, c[1], r[1],
 c[2], r[2], ..., c[n], r[n]):
   For i <- 1..n
      X[i] <- c[i]*A[i]+r[i]*G
      Y[i] <- c[i]*I+r[i]*H(A[i])
   End For
   If H(H(M) || X[1] || Y[1] || X[2] || Y[2] || ... || X[n] || Y[n])
    = Sum[i=1..n](c[i])
      Return "Correct"
   Else
      Return "Incorrect"
   End If
End Procedure
```

Figure 2.12: Ring signature verification in CryptoNote Standard `CNS002` [109].

2.3.3.2 Variable-length Encoding of Integers (Varint)

Varint encodes integers into little-endian base-128 values. The decoding process requires scanning from the first byte to the last byte. The last byte always has a value of less than 128, in which its binary value starts from zero. Varint is beneficial in Monero data serialisation if the system needs to serialise arbitrary large numbers.

2.3.3.3 CryptoNote Blockchain

CNS003 entails blockchain specification in CryptoNote. A block contains three parts, namely block header, base transaction body, and list of transaction identifiers. The specification includes the following items.

- Block header. A block header contains the following information.
  - *major_version*. (varint)
  - *minor_version* (varint)
  - *timestamp* (varint)
  - *prev_id* (32-byte hash)
  - *nonce* (4 bytes).

Note: Although CNS describes *major_version* and *minor_version* as varint fields, in reality, each field only consists of one byte[3].

- Base transaction. A valid block always contains one base transaction. The base transaction in CryptoNote is equal to Bitcoin's coinbase transaction. The base transaction is the first transaction in the block that sends the block reward to the miner that creates the block. A base transaction contains the following information.
  - *version* (varint)
  - *unlock_time* (varint)
  - *input_num* (varint) denotes the number of inputs of the transaction. A base transaction always has its value set to one.
  - *input_type* (byte), valued to $0xff$ in a base transaction.
  - *height* (varint) refers to the block height.
  - *output_num* (varint) describes the number of outputs.
  - *outputs* (array of outputs) is the array of the outputs of the transaction.
  - *extra_size* (varint) defines the size of the *extra* field.
  - *extra* (array of bytes) is an additional data related to the transaction.
- List of transaction identifiers. A transaction identifier is a hash value from Keccak hash function of the actual transaction body. A *tx_num* (varint) informs the number of transaction identifiers, followed by array of identifiers (array of hashes).

A block identifier is a Keccak hash value of the following fields.

- size of the next three fields (varint)
- block_header
- Merkle root hash
- number of transactions (varint)

The block identifier uniquely distinguishes blocks.

CryptoNote also essentially uses a Merkle tree to verify transactions. A Merkle root hash is the root of a binary hash from the transactions that pose as leaves. If

[3]Based on our experience in deserialising block header in January 2020, where the blockchain version is `12.12` or `0c0c`.

```
h[i] = H(h[2*i] || h[2*i+1])
  where 1 <= i <= m-1 or 3*m-n <= i <= 2*m-1.
h[i] = H(tx[i-m])
  where m <= i <= 3*m-n-1
h[i] = H(tx[i-4*m+n])
  where 6*m-2*n <= i <= 4*m-1.
```

Figure 2.13: Merkle root hash algorithm based on CryptoNote Standard `CNS003` [113].

- $tx[i]$ is the i-th transaction of the block and $0 \leq i \leq n-1$,
- n is the number of transaction,
- $tx[0]$ is the base transaction,
- m is the largest power of two, $m \leq m$,
- H is the Keccak function.

then the algorithm to construct $2n-1$ nodes h is shown in Figure 2.13. The Merkle root hash is indicated by $h[1]$.

An example of CryptoNote Merkle tree is in Figure 2.14. CryptoNote uses an unbalanced Merkle tree by adding a new layer for odd-number transactions. The approach is different compared to Bitcoin-style blockchain, where the odd transaction is hashed with itself to get a node.

2.3.3.4 CryptoNote Transactions

CryptoNote Standard 004 [115] defines the CryptoNote transaction specification. A CryptoNote transaction contains two parts, namely transaction prefix and signatures.

A transaction prefix contains several fields as follows.

- *version* (varint)
- *unlock_time* (varint)
- *input_num* (varint)

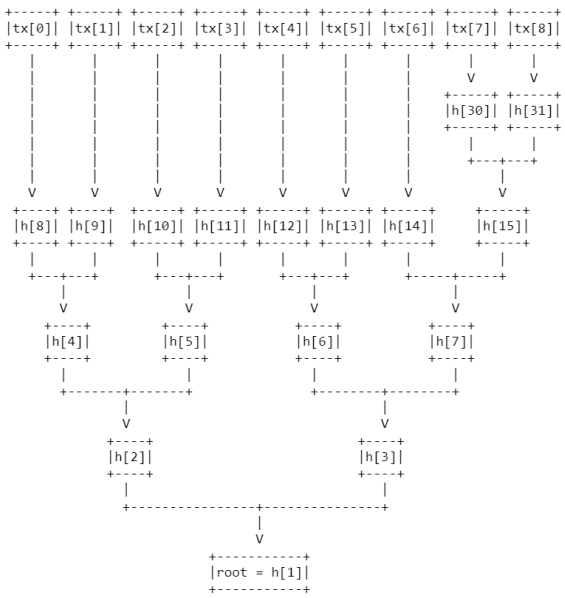

Figure 2.14: An example of a Merkle tree construction in CryptoNote Standard `CNS003` [113].

- *inputs* (varint). There are two types of inputs, namely `txin_gen` and `txin_to_key`. The `txin_gen` can only be used in a base transaction. It contains two fields as follows.
  - *input_type* (byte). A value of `0xff` indicates *txin_gen*.
  - *height* (varint)

  On the other hand, `txin_to_key` is used in other transactions. A `txin_to_key` spends an existing `txout_to_key`. To obfuscate the real spent `txout_to_key`, a ring signature is attached to the second part of a transaction. A `txin_to_key` contains the following fields:
  - *input_type* (byte). A value of $0x2$ indicates `txin_to_key`.
  - *amount* (varint) is the amount of the input.
  - *key_offset_num* (varint) shows the number of keys in the input.
  - *key_offsets* (array of varint) contains the offsets (global indexes) of keys involved in the inputs, also called as mixins or decoys.

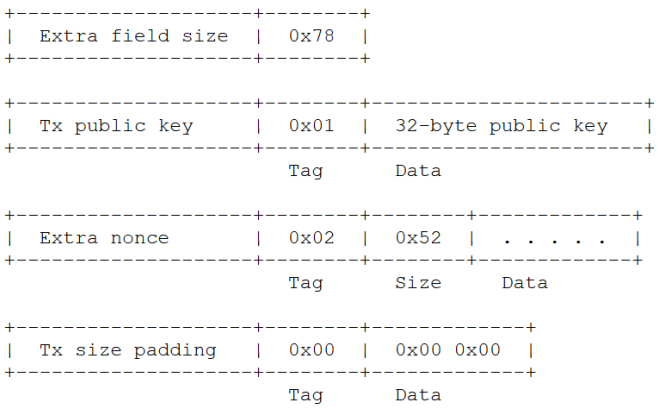

Figure 2.15: Transaction extra field example. [114]

  - *key_image* (key_image type) is the tag that prevents double spending in Linkable Ring Signature scheme. The tag is unique; it can only appear once in its lifetime, which indicates that one coin can only be spent once.

- *output_num* (varint)
- *outputs* (array of outputs). In detail, each output contains two pieces of information as follows.
  - *amount* (varint) defines the amount of the output.
  - *(*target) (output target) that specifies how the receiver can spend the coin. In most cases, the *target* is set to `txout_to_key`. The `txout_to_key` defines that the coin is redeemable by using the owner's private key.
- *extra_size* (varint)
- *extra* (array of bytes) that contains additional information related to the transaction. The *extra* field is used for one-time public key-related operations. `CNS005` further explains *extra* field, which consists of two parts, namely *tag* and *data*. It can also contain *size* field inserted before *data* field if the value of the *tag* field is either `0x00` (transaction padding) or `0x02` (extra nonce for mining pools). If the value of the *tag* is `0x01`, then it has a fixed length of 32 bytes, which is the transaction public key. Figure 2.15 shows an example of transaction extra field.

2.3.3.5 CryptoNote One-Time Keys

The CryptoNote Standard 006 [110] defines a key generation and key recovery schemes. Let Alice wants to send a coin S to Bob's address (A, B). The key generation is as follows.

1. Alice generates a random integer r mod l.
2. Alice computes a *tx_pubkey* $R = r * G$.
3. Let n as the output index (varint) in the transaction, Alice computes a one-time public key $P = H(r * A||n) * G + B$.
4. Alice puts (S, P) in the transaction input.
5. Alice puts R in the *extra* field.

Bob knows the secret keys that corresponds to his address, namely (a, b). Bob needs to recover the key by using the key recovery algorithm. Since Bob does not know which coin was sent by Alice, Bob runs the algorithm on every output of every new transaction.

1. Bob computes $P' = H(a * R||n) * G + B$.
2. Bob compares P' to P. If equal, then Bob knows that the output is his.
3. Bob computes the output's secret key $x = H(a * R||n) + b$.

Figure 2.16 illustrates the key generation and key recovery scheme in CryptoNote.

2.3.3.6 CryptoNote Keys and Addresses

The CryptoNote Standard 007 [55] uses the term keys to refer to two different meanings, namely `txout_to_key` or one-time keys [110] and the users' long-term keys (or permanent keys). The long-term keys consist of two pairs of public and private keys. The wallets store private keys (a, b). The public keys (A, B) become the address. Each CryptoNote key is 32-byte encoding of Ed25519 [10].

CryptoNote also uses two other terms to refer keys based on their purposes, namely spend key and view key. A spend key is a pair of private keys (a, b) that corresponds to an address or public keys (A, B) that enables its owner to sign transactions. A view key is a combination of a private key and a public key (a, B). A view key holder can identify incoming transactions to its corresponding address (A, B), but the holder cannot:

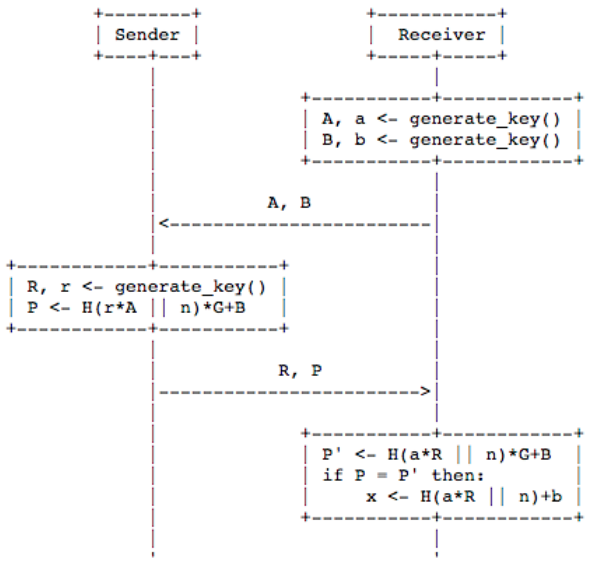

Figure 2.16: Key generation and key recovery in CryptoNote [110].

- identify outgoing transactions
- compute the balance of the wallet
- sign any transactions
- spend any coins

A CryptoNote address uses Base58 as its encoding scheme. An address is a serialisation of three parts, namely:

- *Prefix* (varint) to distinguish addresses of different cryptocurrency.
- *Public keys* (A, B)
- *Checksum* (4 bytes) $C = H(Prefix||A||B)[0..3]$

The address is formulated as follows: $Address = Base58(Prefix||A||B||Checksum)$. The construction is shown in Figure 2.17.

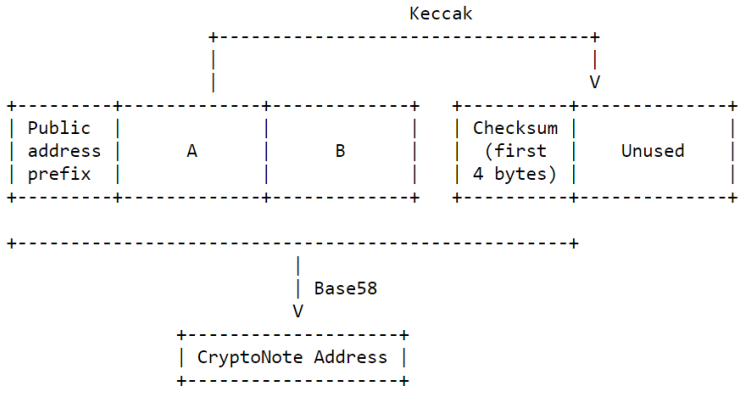

Figure 2.17: CryptoNote address serialisation [55].

2.3.3.7 Base58

Base58 is a popular encoding scheme for cryptocurrency addresses such as Bitcoin. CryptoNote also adopts Base58 for address encoding. It uses a set of alphanumeric characters that will not create confusion. Base58 omits characters such as zero and letter l (lowercase L), I (uppercase i), and O (uppercase o). The characters are as follows:

`123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz` [55]

The encoding scheme splits the input into 8-byte blocks. If the input is not a multiple of eight, then the last block contains less than 8 bytes. The scheme processes each block separately by converting the block into a big-endian integer by using modulus 58 operation, then into a Base58 character. Each block requires 11 or fewer characters.

2.4 Monero

Monero is a CryptoNote-based cryptocurrency. Its codebase is from ByteCoin source code. Similar to other cryptocurrencies, there are applications required to run the Monero system, namely Monero daemon and Monero wallet. A Monero daemon is the full node keeping a full record of all transactions happening in the network, while a Monero wallet does not need to store all transactions locally. Monero wallet helps to create new transactions by connecting to a Monero daemon.

When creating a new transaction, the Monero wallet first sends a request to the Monero

daemon. The purpose of the requests is to get information about potential public keys (outputs) to be used as decoys in the ring signature construction. The wallet selects a set of global indexes from the histogram provided by the daemon by using a random sampling algorithm. Then the daemon provides the corresponding outputs. The index of the real output is also in the request for two reasons: as a test whether the daemon sends the correct outputs and to obfuscate the final ring signature from a curious daemon.

Monero developers have adopted additional protection on users' privacy. The Confidential Transaction (CT) is added into ring signature construction to create the RingCT [83]. The confidential transaction is a Pedersen commitment to encrypting the amount of money sent from the sender to the receiver so that they cannot be visible to the world; only the respective participants can decrypt the amount (Maxwell, 2015). RingCT was deployed in January 2017 and became mandatory since September 2017[4].

RingCT has brought a significant change on the way users create mixins. In a non-RingCT transaction, an output can only mix with other outputs with the same amount. Before RingCT is available, an output having an unprecedented amount of coins cannot mix with other outputs. Hence zero-mixin transaction occurs. The zero-mixin transaction is a transaction containing an input that does not have any mixin. RingCT encrypts the amount of money. Therefore, a RingCT output can mix with any outputs.

2.4.1 Monero-related Applications

There are at least two applications that need to be present in the Monero system: Monero Daemon and Monero Wallet. Monero Daemon handles the main protocols in the system: peer-to-peer network, consensus (mining), storing data to the blockchain and retrieving the data from the blockchain. Monero Wallet, on the other hand, poses as a client by managing private keys owned by the user. The Monero Wallet also helps the user to create new transactions and send the raw transactions to the Monero Daemon. Monero Wallet also identifies new incoming transactions by retrieving the blockchain data from Monero Daemon.

2.4.2 Monero Address

Similar to CryptoNote addressing scheme [55], Monero uses two sets of public keys [108]. The public key cryptosystem in Monero uses elliptic curve of Twisted Edwards curve

[4]https://getmonero.org/resources/moneropedia/ringCT.html

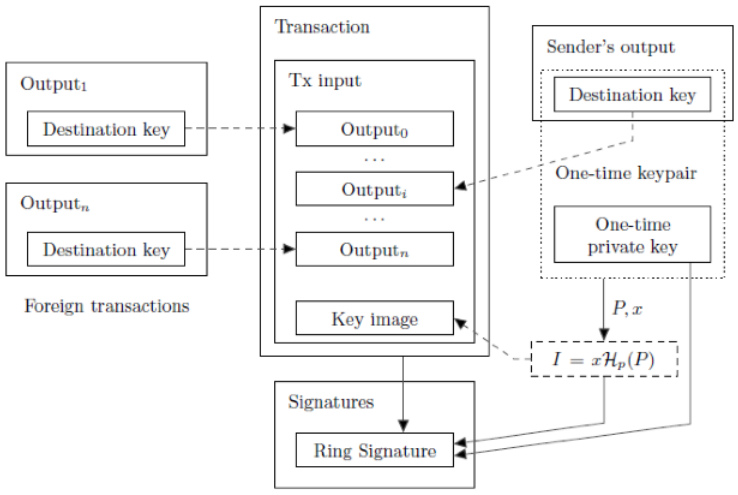

Figure 2.18: The construction of Monero transaction [108].

`ed25519`. The private keys a and b are associated with public keys A and B respectively, such that $A = aG$ and $B = bG$, where G is a base point on the elliptic curve. The address of Monero consists of two public keys A and B, denoted as (A, B). The receiver sends his or her Monero address to the sender, which then the sender could generate the one-time address for the receiver [110].

Monero also provides an audit feature called tracking key or view key, denoted as (a, B). The tracking key is useful to determine all incoming fund to its address. However, the tracking key cannot determine the address balance. It is also not possible to spend the money by using the tracking key since it does not contain the private key b.

2.4.3 Monero Transaction

In Monero, the transaction is using a similar approach as in Bitcoin, which is UTXO-based system. In such a system, a user will need to prove her ownership for each unspent output she wants to spend. Moreover, since Monero uses a one-time public key mechanism, then each unspent output will be associated with a different public key each time [108]. All outputs in the blockchain become a pool of potential mixins. Users can select at random the mixins from the pool to obfuscate their transaction inputs.

Monero transaction as in Figure 2.18 contains several information as follows.

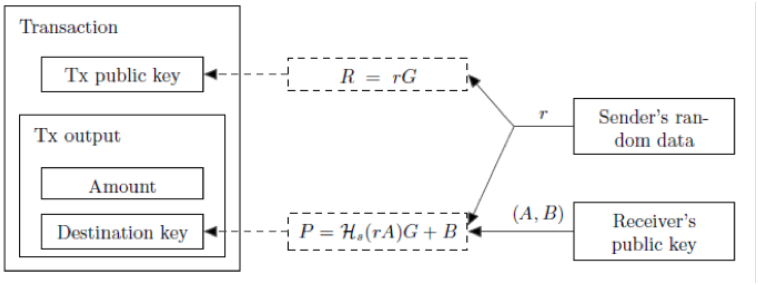

Figure 2.19: A sender constructs the receiver's address [108].

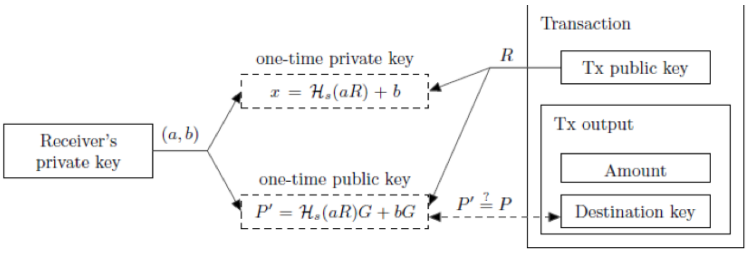

Figure 2.20: The receiver computes the public key [108].

- Groups of outputs as the inputs. A group of outputs contains several public keys; one of these public keys is the real output to be spent, while the other public keys are the decoys.
- Key images. Each group of outputs needs to provide a key image. The key image is the double-spending protection mechanism in Monero. If a key image appears more than once in the blockchain, it means the same output has been double-spent.
- Outputs. These outputs are the new UTXO designated to the payee(s) or the payer as change money.

Diffie-Hellman key exchange guarantees that the receiver can retrieve the new address designated to the receiver. A random value r is picked by the sender, while a transaction public key $R = rG$ is also attached to the transaction along with a destination key $P = H_s(rA)G + B$ as in Figure 2.19. The receiver can calculate the $P' = H_s(aR)G + bG$ for each transaction appearing in the blockchain. If $P = P'$ then the receiver can determine that the money is his or hers, as shown in Figure 2.20.

2.4.4 Monero Payment ID

In CryptoNote-based cryptocurrencies such as Monero, a sender is indistinguishable among a set of senders [108]. However, a merchant must identify the sender of each transaction. In Bitcoin, the merchant can create a unique address for each customer by using a deterministic wallet [123]. However, in the past, Monero only allows one primary Monero address in a wallet (we exclude subaddress [82] in the discussion). Therefore, metadata called Payment ID is added into the Monero transaction to help the receiver to determine the sender of the payment.

The Payment ID is 32 bytes data inserted into the extra field on the transaction data that consists of 64 digits hexadecimal as appears on the Monero blockchain explorer. A new feature named integrated address encrypts the Payment ID into the destination address, where only the receiver can decrypt the Payment ID. The Payment ID for the integrated address is 8 bytes long or 16 digits hexadecimal.

A new feature named integrated address encrypts the Payment ID into the destination address, where only the receiver can decrypt the Payment ID. The Payment ID for the integrated address is 8 bytes long or 16 digits hexadecimal. We use the term `UPID` to refer to the unencrypted Payment ID and `EPID` to refer to the encrypted Payment ID.

2.4.5 Monero Ring Confidential Transaction

Ring Confidential Transaction (RingCT) is a feature in Monero which was first deployed in Wolfram Warptangent (version 5) and has become a mandatory in Helium Hydra (version 6). The first block containing a RingCT Transaction is block number 1,220,517 in the Monero blockchain. RingCT is a technique that combines Monero's Linkable Ring Signature and Confidential Transaction proposed by Maxwell [70]. RingCT combines Confidential Transaction and Ring Signature [83].

RingCT mitigates liquidity problem by hiding the number of coins contained in the public keys. The main requirement for constructing a ring signature is that each ring member must have the same amount of coins and therefore the real one cannot be distinguished from the decoys.

There is a problem of liquidity that the users cannot construct a transaction with enough decoys that they employ zero-mixin transaction [85]. The zero mixin transaction does not have any decoys. Hence, it is traceable because there is 100% chance of guessing the real spent key in the zero-mixin transaction, unlike the ones with decoys. RingCT

transaction outputs will appear as containing zero coins. Therefore the number of decoys can be selected from a large pool of public keys using the same obfuscation method.

2.5 Monero Anonymity Problems

Over the past few years, there have been several studies to explore the anonymity features of Monero. In 2014, Black Marble Attack emerged. Other attack methods such as including zero-mixin, temporal analysis, private view key, EABE attack, and the problem of key reuse reduce the users' anonymity. We discuss each technique in the following subsections.

2.5.1 Black Marble Attack

The term Black Marbles Attack refers to an attack against the Monero anonymity conducted by controlling as many outputs as possible in the Monero blockchain [85]. In the attack, an attacker tries to control as many outputs as possible. Let all outputs of the blockchain as black and white marbles in an urn. Black marbles are the outputs that belong to the attacker. On the other hand, white marbles are the outputs created by honest users. The urn is the shared ledger (the blockchain) where the black marbles (malicious outputs) and the white marbles (honest outputs) are stored and visible to all observers.

To maximise the impact of the attack, the attacker needs to create more outputs (the black marbles) to have more outputs than other users' outputs (the white marbles). The attacker creates as many transactions as possible and sends the coins to her address [65]. Since there is no information about spent outputs, the attacker needs to continually add more outputs to increase the probability of her outputs being picked up by new transactions as decoys.

2.5.2 Zero-mixin Transaction and Cascade Effect

In Monero, zero mixin transactions are a transaction which has at least one input using no decoys or mixins. Zero mixin transactions do not have any anonymity feature offered by ring signature, and therefore, any observers can immediately trace the real sender of the transaction. The anonymity problem does not occur only for the zero mixin transactions, but also for every other transaction that uses the same outputs as their decoys. The ring signature is an effective method to create plausible deniability for untraceability under an

optimum environment, namely there exist enough outputs that share identical characteristics such as age and amount of coins contained in the outputs). Unfortunately, this environment could not be sufficiently provided by Monero before the release of the mandatory RingCT usage.

Although the users must split their transactions according to a specific denomination regulation, this regulation was never strictly applied. An unprecedented amount of coins can still be confirmed in the transaction, although it will create a liquidity problem where the user cannot find other outputs containing the same amount of coins. For all outputs that cannot mix with any other outputs, the users create a zero-mixin transaction: an input contains only the real output without any mixin or decoys.

Although the zero mixin transactions have a cascade effect on the anonymity of other transactions [85, 65], new investigations show that the impact is more significant than expected. Based on techniques presented in the previous research, the effect is reaching the rate of 87% [59, 79]. It means that at least more than half of all analysed inputs (before RingCT) are distinguishable between the decoys and the real spent outputs.

2.5.3 Temporal Analysis

The zero-mixin transaction analysis also reveals that in most cases, the real outputs are the most recent outputs [79]. The extrapolation of the gathered data mentions that 80% of the real outputs that can be detected are the newest. The uniform mixin sampling used by the system could not hide this characteristic.

New sampling methods, triangular distribution, were introduced to tackle the problem. Triangular distribution protocol describes that at least 25% of all decoys must be from recently added outputs. By using this technique, the temporal analysis becomes ineffective because at least one in the mandatory minimum of five mixins in the version 6 of Monero (Helium Hydra or software version 0.11.0) is aged less than five days.

2.5.4 Publishing Private View Keys

The private view key is a feature within the Monero system to provide an auditability of the coins owned by a user. Assuming that the user provides the private view key of her wallet to an auditor, the auditor is then able to track every coin received by the associated address. Although the private view key enables such thing, the auditor cannot steal the coins from the user by using the private view key. It is also impossible for the auditors

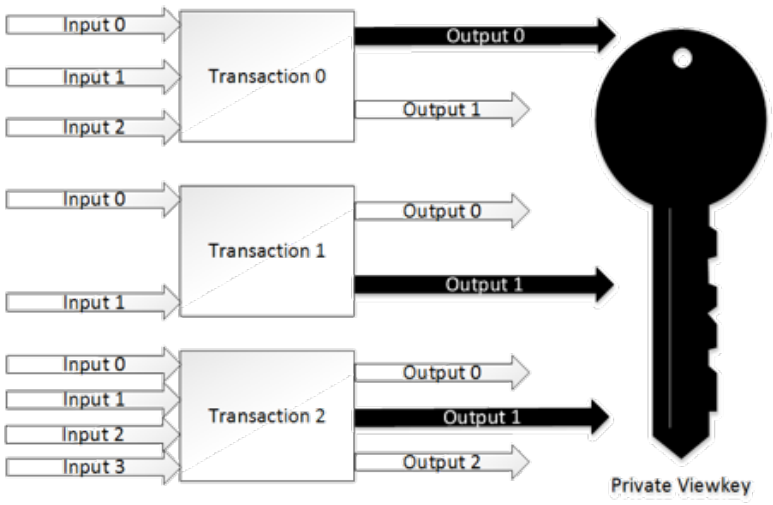

Figure 2.21: A private viewkey determines all incoming outputs (payments) sent to the corresponding address by scanning all transactions in the blockchain.

to determine the address balance. The high-level construction of a private view key is in Figure 2.21.

The private view key can become a risk to Monero unlinkability. The anonymity of a user depends on the anonymity of other users. The private view keys can distinguish between the outputs sent to the owner of the private view keys and the outputs sent back to the sender (the change).

The private view key is also an important feature. It is usable for use-cases that require compliance, such as auditing and charities[5]. However, after the audit, it is not possible to regain the unlinkability feature without creating a new address and move all the balance to the new address.

2.5.5 EABE Attack

EABE attack is a scheme where the existence of cryptocurrency exchanges can disrupt the anonymity of the user [58]. EABE is an abbreviation of *Exchange-Alice-Bob-Exchange*. The scheme works as follows. Suppose Alice buys her Monero from a cryptocurrency exchange. In order to use the exchange, Alice needs to provide her ID card to the company. Hence, all Monero coins she buys are identifiable to her real ID.

[5]According to Moneroblocks' Monero Richlist https://moneroblocks.info/richlist

Later, she transfers her coins to Bob for payment of products she purchases from Bob. Bob needs fiat money. Therefore he sells the Monero he receives from Alice to an exchange (or the same exchange platform as Alice). Bob also needs to provide his ID card to the cryptocurrency exchange company. Hence the coin is also connected to his ID. If Alice and Bob are using the same cryptocurrency exchange, then the company will be able to trace the transaction from the beginning to the end.

2.6 Summary

In this literature review, we first discuss fundamental blockchain technology in Bitcoin. Then, we discuss how its privacy issues lead to the development of privacy-preserving techniques in Bitcoin, and later, completely independent privacy-preserving cryptocurrencies such as Monero. We discuss technical details in Monero that supports anonymous transactions such as Linkable Ring Signature (`LRS`) and One-time Public Key (`OTPK`).

We also provide detailed explanations of various CryptoNote Standards (CNS) [113, 115, 114, 110, 55] that extends CryptoNote whitepaper [108] as the underlying Monero protocols. We cover CryptoNote's blockchain data structure, transaction structure, data types, and encoding scheme. We also explain the implementation of CryptoNote's theoretical concepts into Monero working system and how they fit into Monero. We also discuss recent technology development in Monero that improves transaction anonymity, most notably Ring Confidential Transaction (`RingCT`), that encrypts transaction amount.

We discuss known Monero anonymity problems such as black marble attack [85], zero-mixin transaction and cascade attack [59, 73, 79], temporal analysis [59], private view key publication, and EABE attack [58]. The known problems become the starting point of this project to investigate potential threats to Monero anonymity further.

2.7 Research Gaps

After conducting the literature review, we identify the research gaps in anonymity in cryptocurrency. The privacy-preserving technique developments mainly focus on the underlying cryptography. While the cryptographic methods are provably secure against known attacks, cryptographic implementations into working systems are understudied. Recent research indicates potential protocol-level vulnerabilities in Monero, one of the privacy-preserving cryptocurrencies.

Chapter 3

A Transaction-based Attack to Monero Anonymity

Monero is one of the privacy-preserving cryptocurrencies employing CryptoNote protocol. Cryptographic techniques such as linkable ring signature and one-time public key support privacy feature in Monero. Recent studies show that the majority of Monero inputs are traceable. However, RingCT solves the identified problem. Monero ring signature selects current outputs in the blockchain as ring members. A user does not need an output owner's permission to include the output in the user's transaction. Monero developers recommended a set of output selection mechanisms; however, the transaction creator determines how he or she selects the outputs for transaction decoys. Transaction anonymity depends on the quality of the selected decoys.

This chapter is the first part of our exploration of the first research question, `Q1`, about how attackers can exploit the Monero transaction mechanisms to reduce the anonymity of Monero users. We examine past attacks in Monero and whether they are still applicable to the current system (at the time of writing). We then review the Monero transaction mechanism.

We propose a novel attack to reduce the anonymity of Monero transactions or even to fully deanonymise the inputs. The proposed protocol can be launched in RingCT protocol and enables multiple attackers to collaborate without trusting each other. Monero services such as exchanges and wallets can implement the attack without extra fees and without putting the users' money at risk.

We construct this chapter based on our published paper, *Monero Ring Attack: Recreating Zero Mixin Transaction Effect* [116].

3.1 Introduction

Monero is one of the most valuable privacy-preserving cryptocurrencies in the world. It is built based on blockchain technology where the transaction data is visible to everyone, similar to Bitcoin [81]. However, unlike in Bitcoin, where anyone can track the flow of money between addresses, in Monero, the observers cannot do the same. Ring signature and one-time public key technologies are implemented as the default settings to improve the anonymity of the transaction data. The real senders are obfuscated by adding multiple decoys where the set of possible senders are equal and indistinguishable among each other. The one-time public key means each output is unique and freshly created, while the real address of the receiver never appears in the blockchain. Without any additional information, it is infeasible to determine which addresses belong to a specific user.

Although Monero implements several privacy-preserving methodologies, there are at least four different analyses that researchers developed to reveal hidden information in Monero environment. These analyses were successfully conducted due to the transparency of the blockchain data, liquidity problem, and identified users behaviour.

We propose a novel attack against the Monero untraceability. The proposed attack can reveal the real outputs being spent in Monero transactions or at least reduce the anonymity of the inputs. The attack scheme is available for a single attacker or multiple attackers colluding to launch the scheme without the need of trusting each other. Each attacker will take the benefit of others' results. Our attack is sufficient to be conducted in RingCT environment where an observer cannot see the transaction amount. Constructing a malicious transaction as described in the proposed attack scheme will not cost an extra fee if the attack is attached to an existing service.

3.1.1 Threat Model

We define the threat model in Monero as follows. Everyone can see all information stored in Monero blockchain. The security of the confirmed transactions depends on the consensus model of Monero. We also define two attackers, Attacker A and Attacker B. The Attacker A has a sufficient fund to create transactions but does not have any access to coin exchanges or wallet services software.

There exists a set of attacker $A = \{A1, A2, A3, \dots\}$ colluding to attack the system, but they do not trust each other. The attacker B has all ability owned by the Attacker A. Attacker B can modify coin exchanges or wallet services software. There also exists a set of Attacker $B = \{B1, B2, B3, \dots\}$ colluding without trusting each other. The Attacker A and

Attacker B can also collude to get the best result out of their efforts. Attackers conduct all phases in the proposed method.

3.2 Our Proposed Attack

3.2.1 Overview

The proposed attack scheme utilises the leniency of the Monero daemon towards the transaction created by the Monero wallet. Monero daemon only checks the validity of the transactions submitted. Those transactions require correct balances and valid digital signatures. The wallet entirely processes ring construction. Monero daemon helps the wallet by providing public keys information based on indexes selected by the wallet.

Based on the Monero protocol as described, it is possible to construct a malicious transaction to reduce the k-anonymity or even de-anonymise Monero transactions. We will show that the impact of the proposed attack is similar to the cascade effect from the zero mixin transaction.

We denote a Monero transaction $T_i := R_i \rightarrow O_i$ as an i-th transaction in Monero blockchain B that inputs a set of rings R and outputs a set of O. A ring $R_i^j \in R_i$ is a ring signature that consists of a set of r_i^j existing outputs $\{o_i^1, o_i^2, \dots\}$. The cardinality of R_i^j is r_i^j, or $|R_i^j| = r_i^j$. We denote r as the system's minimum ring size, such that $r \leq r_i^j$. In our scheme, the ring R_i^j spends o_i^k. The output o_i^k is associated to a unique key image I_i^j, where the association is only known to the owner. A ring in a new transaction reveals a unique key image that has never appeared in the blockchain system B that records all known key images as I.

Our attack's preparation phase generates a set of r transactions $\tau = \{T_i, T_j, \dots\}$, where $i, j, ...$ identify the transaction indexes in the blockchain B. Note that the transaction sequence of each $T \in \tau$ do not matter. We assume each transaction contains one identical ring with r ring members, namely R_l, where a ring $R_l^m = R_l$ reveals a unique key image I_l^m during the setup phase. Since there are r valid transactions, it means all ring members are spent by the set of transactions τ. The passive attack phase observes the blockchain B for a transaction T_v that includes q outputs in R_l, $q \leq r$, in one of the rings, R_v^u. We denote ring R_v^u suffers q anonymity reduction. In our active attack, a compromised service constructs a malicious transaction such that $q = r - 1$, where observers are able to identify that there exists exactly one ring member that satisfies $o_v^w \in R_v^u$ and $o_v^w \notin R_l$, which implies that the key image I_v^u is the secret value of o_v^w.

In the detailed explanation of the proposed attack, we also use the term public key to refer to an output to avoid confusion. The phrase public key is correct in Monero, since Monero uses one-time public key scheme, where each output represents a unique public key that can only appear once in a lifetime.

3.2.2 The Proposed method

The proposed attack consists of three phases: preparation, setup, and attack. The attack phase has two different approaches, the passive and active attack. The following subsections explain each attack approach.

3.2.2.1 Preparation Phase

For each thread of attack, the attacker needs to have many unspent outputs. The number of outputs depends on the minimum ring size r required by Monero system. For version 6 (Helium Hydra), the minimum ring size r is five and therefore the minimum number of outputs required by the attacker is equal to that number. The purpose of the preparation phase is to have a set of unspent outputs in which every single output is spendable in the setup phase. If the attacker has more unspent outputs than the minimum ring size but less than multiples of r, then the remaining outputs can be used during the attack phase.

Since the deployment of RingCT, it is not necessary to have the same amount of coins for each output. Therefore, an attacker can use a small number of coins split across multiple outputs. It means that an attacker can only focus on paying the transaction fees and do not need to have extra reserved coins.

The type of attack influences the attacker's number of threads. If the attacker intends to launch a passive attack, then the attacker needs to create as many threads as possible. The success of the attacker depends on the number of threads created by the attacker, whereas in the active attack, the attacker only needs to create one thread. The outputs are reusable in future transactions, which will not reduce the effectiveness of the attack, although it might raise suspicion if a specific output appears too many times.

3.2.2.2 Setup Phase

In the phase, each attack thread requires exactly r inputs. It means there will be r rings created by the attacker. Each ring will spend a transaction output owned by the attacker.

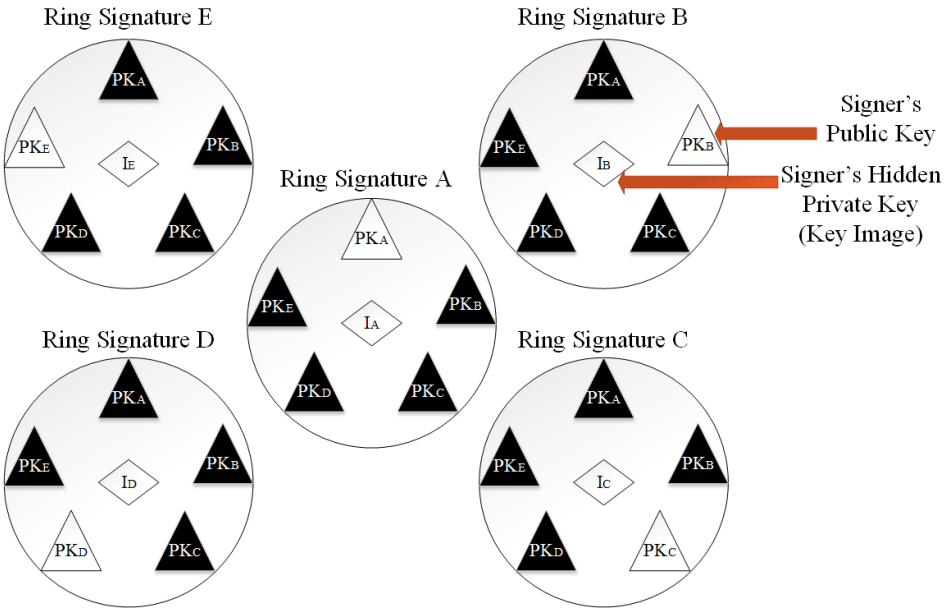

Figure 3.1: The Setup Phase Where $r = 5$.

Let a set of l public keys $L = \{PK_A, PK_B, PK_C, PK_D, PK_E, \dots\}$ and their secret key image pairs $K = \{I_A, I_B, I_C, I_D, I_E, \dots\}$. The number of public keys in L is equal to r. The decoys for each ring is chosen from L, as shown in Figure 3.1.

The attacker can generate one or multiple transactions to spend the inputs. However, it is more cost-effective to have r inputs all in the same transaction. The setup phase has a similar effect as the zero-mixin transaction, with one difference. In the zero mixin transaction, anyone can precisely determine which input spends an output. In this setup phase, it is infeasible to determine the exact input that spends a particular output. We can only say that all inputs in the setup phase spend all members of L, regardless of which input spends which output. The focus of the setup phase is to nullify the probability of other transactions spending any member of L.

If the attacker does not create r inputs where all of the outputs are the member of L, then the requirement to recreate the zero-mixin transaction effect is not fulfilled. Other observers cannot detect whether the transactions spend the inputs, and therefore the attack phase cannot be conducted.

3.2.2.3 Attack Phase

There are two types of attack, namely, passive attack and active attack. Each attack has different purposes and different methods. The following describes both attack types.

Passive attack. The purpose of the passive attack is to allow the outputs spent (members of L) in the setup phase to be used by other users in multiple transactions. If any members of L becomes a decoy, the observer can omit the public keys as setup phase transaction already spent the public keys. According to double-spending prevention rule, it is not possible to re-spend the public keys in any other transactions. The related transactions suffer a reduced k-anonymity. The degree of the reduced k-anonymity depends on the number of decoys coming from the transactions in the setup phase. The example in Figure 3.2 depicts a case with reduced anonymity by two, according to the number of spent public keys used as decoys.

Active attack. Let there be a malicious Monero wallet service run by Attacker B. The purpose of the wallet is not to steal the coins owned by the users but to make the transactions traceable. The wallet knows the public keys L and use them as decoys in the ring signature as in Figure 3.3.

The active attack is efficient when targeting others' outputs, primarily when a wallet implements the attack protocol. The user might not be able to determine the malicious behaviour of the wallet, as long as the wallet successfully creates transactions.

There are differences between the passive attack and the active attack. In the passive attack, the attacker analyses the result of the setup phase and collects information about other transactions which suffer reduced anonymity. In the active attack, the attacker can altogether remove the anonymity of the inputs. In the passive attack, related services such as coin exchanges or wallet services are not necessary to be used by the users, while in the active attack, an attacker must compromise any related services.

3.2.3 Proof of Concept

As a proof of concept of our proposed attack, we conducted the preparation phase and setup phase. We modify the source code of Monero to create a malicious wallet with the ability to create transactions which comply with our scheme, in particular `simplewallet.cpp` and `wallet2.cpp`.

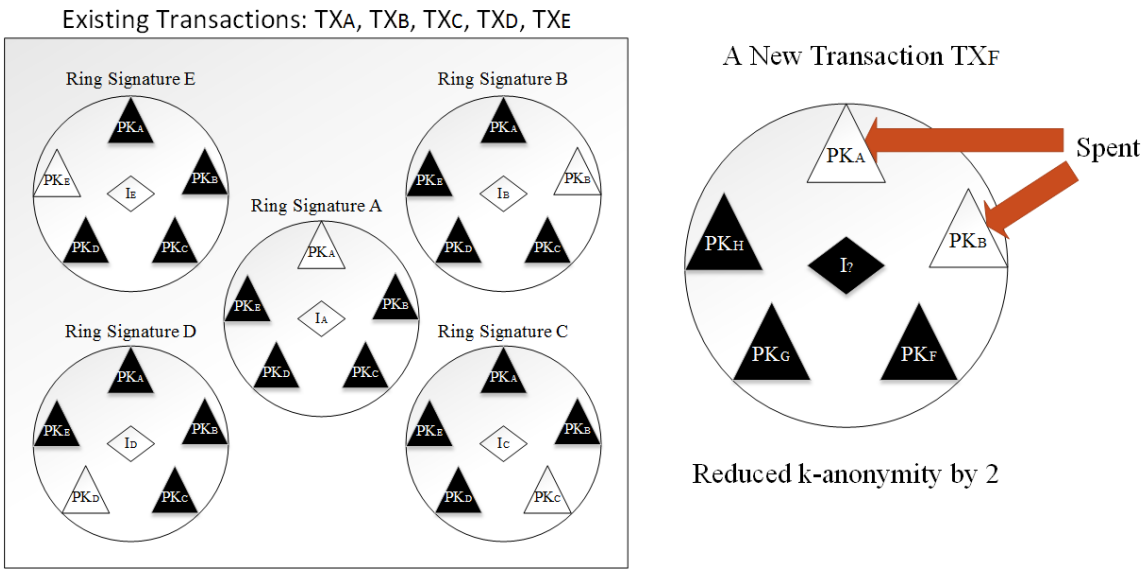

Figure 3.2: The Passive Attack.

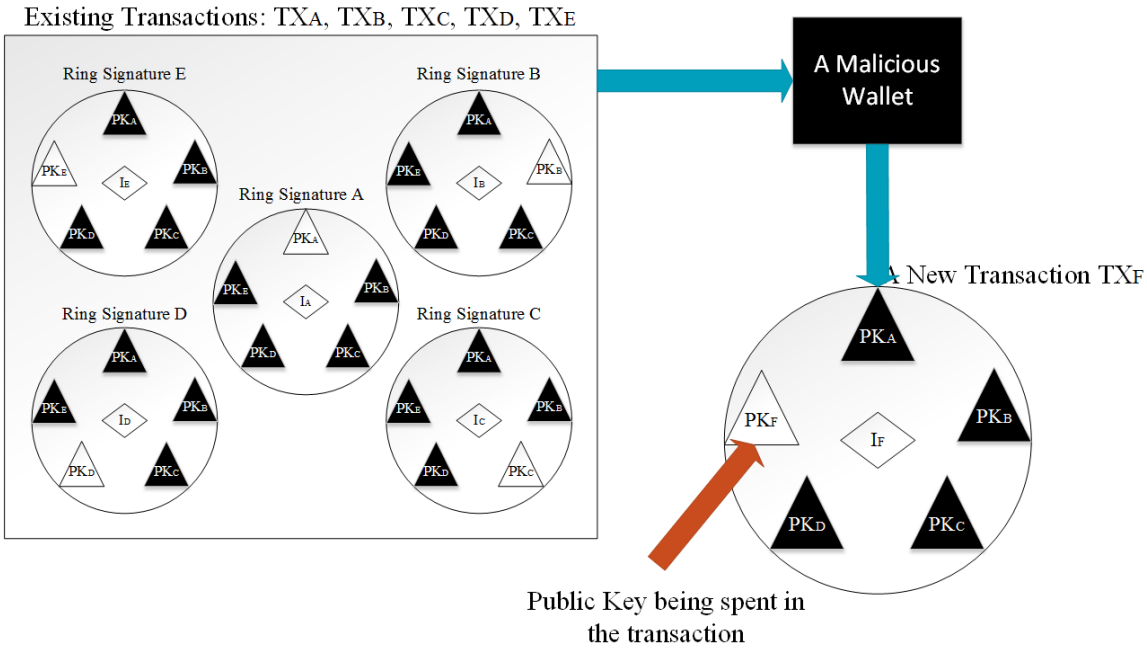

Figure 3.3: The Active Attack.

3.2.3.1 Preparation Phase

Instead of having a normal protocol when picking indexes from the histogram data, we pick the indexes from the public keys stored in our wallet. Therefore, as the wallet also stores the global index for each output, it is not necessary to create any requests to the daemon for the data as the wallet itself can supply all requirements.

We have successfully launched the preparation phase and setup phase into the Monero mainnet. The setup phase consists of one thread with $r = 5$. The transaction ID for the preparation phase is:

`b6781f2a6f5608553546442b84888346fdc3f78dd8995170180ed74081c05362`.

3.2.3.2 Setup Phase

We have executed the setup phase with transaction ID of the following:

`8d4a0c7eccf92542eb5e1f09e72cc0d934b180b768bc95388d33051db83194bb`.

The detail of the transactions is accessible through any online Monero blockchain explorers such as `Xmrchain.net`.

3.2.4 Our Proposed Attack Compared to Existing Attacks

Compared to the Black Marble Attack, our proposed attack is better in a coordinated attack scenario of multiple attackers. Each attacker conducts the attack, and other attackers can utilise the attack results without any additional information such as private view keys. In our proposed attack, the transaction does not require the attacker's address as the receiving side. Therefore, the attack is easy to implement in existing online services such as exchanges or wallet services. The exchanges or the wallet services do not need to spend any extra transaction fees since they only need to implement the attack mechanism into the system and convert their regular transactions into malicious transactions where only the anonymity of the transactions suffers a reduction.

Our proposed attack does not rely on the existence of zero mixin transaction, which becomes obsolete when Monero implemented RingCT protocol and mandatory minimum number of mixins. The setup phase recreates the similar impact of the zero mixin transaction, and the attack phase recreates the cascade effect of zero mixin transaction.

Our proposed attack is even more useful to be launched in RingCT-enabled system because an attacker does not need to attack multiple coin denominations and only focus on

Table 3.1: Comparing attack methods

Factors	BM	ZM	TA	PPV	Ours
Collaboration between attackers	X	V	X	X	V
Requires no extra fees	X	V	V	X	V
RingCT resistant	V	X	V	V	V
Minimum mixin resistant	V	X	V	V	V
Accuracy in determining real outputs	V	V	X	V	V

one denomination. Moreover, RingCT enables the attacker to use a small number of coins. The system cannot detect the number of coins sent, and therefore even the attacker sends zero coins, the system will still accept it.

Our attack has a higher precision rate compared to temporal analysis. The temporal analysis depends on decoy sampling method. If the sampling algorithm is optimum, then temporal analysis cannot determine the real output of the input. Our attack can precisely determine the real output in the input with 100% accuracy. Table 3.1 shows the summary of the comparison.

3.3 Conclusion and Future Work

We have proposed and demonstrated a new attack against the untraceability of Monero system. We showed that the anonymity of Monero relies on wallet implementation and the construction of each transaction. Our attack explores the weakness of the ring signature assumption where the sampling algorithm for the mixins is assumed to be random.

Malicious wallets can break the users' anonymity without necessarily stealing the money that belongs to the users. Detecting such activities require efforts by scanning all existing combinations of the ring construction ever existed in the blockchain, and it is unlikely that unaware users detect such activity as long as they do not lose their money.

Based on our research, it is vital to enhance the protocol to protect the anonymity of the users without trusting the wallet, since the wallet can construct a transaction which will reduce or eliminate the anonymity of the transaction. Detection and blacklisting are two alternative methods to avoid such an attack.

Our proposed attack method is also applicable in other systems that utilise a ring signature scheme, such as electronic voting (e-voting). Such a system suffers our attack in

the following scenario. Suppose there is an election where multiple candidates compete for a position in the government. A candidate wants to buy votes from the voters in order to win the election. Since the e-voting is employing a ring signature, the candidate buys the votes in bulk. In order to do this, a vote-seller coordinator is necessary. The coordinator lists all vote sellers and creates groups of these sellers based on the ring size n used by the e-voting system. Each group consists of n vote seller. Then, all group members inform their public keys that perform as decoys for other members. Each group constructs n transactions having identical ring members to cast votes for the candidate. The transaction construction resembles our proposed setup phase.

3.4 Chapter Summary

In this chapter, we address our first research question, `Q1`, that investigates Monero transaction schemes. We propose Monero Ring Attack (`MRA`) as a new attack scheme at the protocol level. Our research shows that the lack of regulation in protocol implementation leads to anonymity threats.

In the next chapter, we present the second part of our investigation related to our `Q1`. We evaluate `MRA` and propose a mitigation strategy to prevent the attack. We then improve our `MRA` technique and propose Monero Ring Attack (`MRAE`) that is immune to the mitigation strategy for `MRA`.

Chapter 4

Advanced Transaction-based and Payment ID-based Attacks to Monero Anonymity

Monero is one of the most valuable cryptocurrencies in the market, focusing on users' privacy. The built-in features in Monero help users to obfuscate the information of the senders and the receivers, hence achieve better privacy compared to other cryptocurrencies such as Bitcoin. Previous studies discovered multiple anonymity problems within Monero systems and based on these findings, Monero developers improved the system. However, after the improvements, we discovered new attacks to anonymity reduction in Monero system.

This chapter contains the second part of our investigation on our first research question, `Q1`, where we propose two attacks. To construct the first attack, we revisit our `MRA` and study its attack characteristics. For our second attack, we revisit our data extraction results and discover a repetitive pattern useful for an anonymity attack.

Our first attack is an extension of a known attack called `Monero Ring Attack` (`MRA`). We call our new attack as `Monero Ring Attack Extended` (`MRAE`). `MRAE` is a more robust attack that provides identical attack results as `MRA`. Our new attack is immune to `MRA`'s mitigation strategy by avoiding ring duplicates and utilises unique ring combinations. The second attack exploits Unencrypted Payment ID to discover the real transaction inputs. We find more than 332K traceable inputs. The finding is the largest traceability attack result after RingCT implementation.

This chapter is based on our published paper, *Anonymity Reduction Attacks to Monero*

[117]. In response to our findings[1], Monero implements `blackball`[2] function that prevents our attack scheme. Monero developers also phased out Unencrypted Payment ID (UPID) and replaced it with Encrypted Payment ID (EPID, or also called as integrated address) and subaddress [29].

4.1 Introduction

The anonymity in Monero is based on the information obfuscation of the real senders and the real receivers, making it hard for observers to make a direct relationship of the senders and the receivers. The anonymity features of Monero depends on ring signature and stealth address in the system. The privacy-protection mechanism received a further enhancement by RingCT technology, which obfuscates the number of coins transacted.

We define an anonymity reduction attack as an effort made by a malicious user (or an attacker) who tries to de-anonymise other users' transaction. This attack is made by creating transactions and breaking the attacker's anonymity. An example of this attack is Monero Ring Attack [116]. The attack mentioned earlier works on both pre-RingCT and post-RingCT environment and does not rely on zero-mixin transactions [59, 79].

Contributions. We summarise our research contributions as follows.

1. We propose a mitigation strategy on a known Monero anonymity attack exploiting the users' freedom when creating mixins [85]. Our mitigation strategy uses a list of hash values of existing mixins. New transactions are not allowed to use any mixins that have existed in the system, in which their hash values are on the list. By employing the mitigation strategy, future attacks using the same method are preventable entirely.

2. We propose an extension of Monero Ring Attack (MRA) [116]. Our new scheme achieves the same goals of MRA but nullifies the mitigation strategy we designed. The new scheme provides an obfuscation method when choosing the mixins. Hence identical mixins are no longer needed.

3. We propose a solution to avoid the new attack we propose. Monero developers have been working on a new blacklisting mechanism called blackball which will blacklist all known corrupt outputs to mitigate an attack over key reuse[3]. In our solution,

[1] https://github.com/monero-project/monero/pull/3671
[2] https://github.com/monero-project/monero/pull/3428
[3] https://github.com/monero-project/monero/pull/3322

we developed a new metric as a quantitative measurement towards the suspicious level of anonymity reduction attack. The metric is useful to complement the existing blacklisting method, either by implementing it in the Monero core soft-ware or as a separate service.

4. We propose a novel anonymity reduction attack by utilising extra information called Payment ID (PID) in Monero transactions. The scheme enables the attacker to trace the real outputs spent in the transactions. We also suggest a possible countermeasure on the attack.

Organization. The rest of the chapter is organized as follows. Section 4.2 describes previous studies related to our research. In Section 4.3 we propose a mitigation strategy of a known attack, while in Section 4.4 we improve the known attack by removing the weaknesses and propose a stronger attack trait. Our security analysis, attack mode, attack variants, detection, and mitigation method are presented in Section 4.5, 4.6, 4.7, 4.8, and 4.9 respectively. Lastly, Section 4.10 presents a novel attack related to Payment ID usage.

4.2 Related Work

In this section, we recall related work, namely zero-mixin problems and Monero Ring Attack, that are important to help understand the ideas we present in this the chapter.

4.2.1 Monero Zero-Mixin Problems

In general, a receiver of a transaction can determine that her outputs become decoys on other transactions. Therefore, a user may try to attack the system by creating a large number of outputs in order to reduce the anonymity of other transactions [85]. In the attack, the attacker needs to pay a huge amount of transaction fees. The attacker also needs to keep creating new transactions if she wants to control a majority of the outputs available in the network [85]. Recent studies show that zero-mixin transactions have impacted the anonymity of other transactions. Research finds that at least 87% of the mixins up to a point were de-anonymised [59]. Another research proposes a change in the way the Monero wallet samples the mixins by prioritising new outputs rather than randomly picks the mixins from all possible options [73].

4.2.2 Monero Ring Attack

In Chapter 3, we propose a new type of attack targeting Monero anonymity called Monero Ring Attack [116]. This attack is an improvement over the previous attack presented in [85], where an attacker tries to dominate the available outputs in the system. In Monero Ring Attack, several attackers can join forces to attack the system, each by crafting special transactions [116].

If an attacker creates a malicious transaction, then other attackers can use the result of that transaction in addition to their results. Therefore, in summary, each attacker will pay fewer transaction fees, but with a more significant impact. During the attack, the attackers do not need to trust each other or communicate with each other. All they need to do is scanning the blockchain data and determine the malicious transactions created by others.

The critical point on this attack is using identical mixins on malicious transactions. Let r as the minimum number of mixins in the system; an attacker needs to have at least r outputs or sets of r outputs. Then, the attacker constructs the inputs by using r outputs as the mixins. These constructions help any attackers to determine that existing transactions spent all the outputs used in the mixins. Since double-spending is not possible, the attackers know that the outputs are decoys.

The Monero Ring Attack has three phases: `preparation phase`, `setup phase`, and `attack phase`. In the `preparation phase`, the attacker prepares r outputs. In the `setup phase`, the attacker constructs spending transactions using identical mixins, each mixin has r outputs as the ring members. Then, in the `attack phase`, the attacker expands the outputs (active attack) and analyse the result of the attack (passive attack).

The purpose of the attack is to trace the real outputs spent by mixins or at least reduce the *k-anonymity* of the transaction mixins. The attack creates multiple malicious transactions. Then, the attacker expects that the outputs of the attack become mixins of honest transactions.

4.3 Mitigating Monero Ring Attack

4.3.1 Overview

The setup phase in Monero Ring Attack as described in [116] has a unique characteristic where identical mixins appear multiple times. By determining this characteristic, the attack is detectable. The detection is done by hashing all mixins in the blockchain then search for

any hash duplicates. If any mixin duplicates appear, then it indicates that the attack has occurred.

4.3.2 Detection Method

We propose a method to detect whether the attack has occurred in the blockchain. Scanning and blacklisting steps of detecting the attack are as follows.

1. For all mixins in the blockchain $M = \{m_0, m_1, m_2, \dots m_n\}$, produce a set of hash values $H = \{h_0, h_1, h_2, \dots, h_n\}$ by using hash function FH such that $h_0 = FH(m_0)$. A mixin m_0 is a list of outputs $mo_0 = \{ov, ow, ox, oy, oz, \dots\}$ where $v, w, x, y, z, \dots$ are the output indexes.

2. For all mixins M, compute the corresponding ring size $R = \{r_0, r_1, r_2, \dots, r_n\}$ such that r_0 is the ring size of mixin m_0.

3. For all hash values $h_i \in H$, compute the number of occurrence $u \in U = \{u_0, u_1, u_2, \dots, u_n\}$ such that u_i is the number of occurrence of h_i.

4. For each mixin m_j where $0 \leq j \leq n$, if $r_j = u_j$ then the mixin m_j is considered as an attack. All outputs in mo_j needs to be included in a blacklist B.

5. For each mixin mk where $0 \leq k \leq n$, check if mo_k contains any outputs from B. If yes, then add mo_k to the blacklist B.

The blacklist B, as the result of the detection method provided above, could then be published. All of the outputs in the blacklist B are discouraged to be used when sampling outputs for creating mixins.

4.3.3 Mitigation Strategy: Forbid Mixin Duplicates

The Monero Ring Attack has a characteristic of using mixin duplicates when launching the attack. In order to countermeasure this type of attack, Monero daemon can be equipped by a mechanism to reject new transactions having identical mixins. Furthermore, the daemon can maintain a list of mixin hash values that have appeared in the system. New transactions need to prove that the hash values of their mixins have never existed in the blockchain. Considering that creating new mixins is easy, users can resubmit rejected transactions after revising the duplicated mixins.

Table 4.1: Comparison between the existing Monero Ring Attack and our proposed attack

Parameters	MRA	**MRAE**
Attacking untraceability and anonymity	v	**v**
Cooperation between attackers without trust	v	**v**
Undetected using hash table	x	**v**

4.4 Extending Monero Ring Attack

4.4.1 Overview

We propose an extension to the idea of using sets of transactions and creating identical mixins in the setup phase, as explained in Section 4.2.2. Instead of using identical mixins, our new attack uses combinations of outputs. The result of this modification is identical to the Monero Ring Attack, but the method is harder to detect. The hashing detection method is useless against the new attack. Table 4.1 shows the comparison between the Monero Ring Attack (MRA) and our proposed attack.

Both attacks are useful when launched against specific targets, such as coin ex-change users, rather than targeting random Monero users. Targeting random users requires a massive amount of money to pay the transaction fee while targeting specific users will reduce the attack cost. The governments or regulators can enforce business entities under their jurisdictions to implement this scheme in order to discover the users' activities in Monero system secretly.

4.5 Security Model

The security model is similar to the one proposed in MRA [116]. We assume that an attacker has read-only access to the public blockchain. The attacker might also have access to wallet services or trading platforms in order to launch the attack without paying transaction fees. In the mentioned services, customers pay the transaction fees.

The goal of the attack is to define the real outputs spent by other transactions or reduce the anonymity of the transactions created by the users. The attack enables multiple attackers to analyse the work of other attackers and aggregate the result to maximise their efforts.

Table 4.2: Spending six outputs in six inputs. The rasterized cells are the ones being spent.

Outputs	I_A	I_B	I_C	I_D	I_E	I_F
O_1	v		v	v	v	v
O_2	v	v		v	v	v
O_3	v	v	v		v	v
O_4	v	v	v	v		v
O_5	v	v	v	v	v	
O_6		v	v	v	v	v

4.6 Attack Mode

Table 4.2 is an example how 6 outputs $O = \{O_1, O_2, O_3, O_4, O_5, O_6\}$ can be spent in six inputs $I = \{I_A, I_B, I_C, I_D, I_E, I_F\}$, with each input has a ring size of 5. We assume the system's mandatory ring size is five as in Monero fork version 6 (software version v0.11.0.0). The rasterised cells indicate the real outputs of the inputs. The attack can be obfuscated further by employing a large set of outputs O which will be spent by a large set of inputs I. These inputs can be in one or many transactions.

We denote by nCr the number of possible ways of choosing a ring of size r out of a total of n possible outputs. For example, a case where $n = 80$ and $r = 5$, we find $C = 24,040,016$ possible combinations to spend 80 outputs in 80 inputs. The calculation shows that it is infeasible to determine the attack by using a trivial mechanism.

4.7 Collaborating with Other Attackers

The proposed attack can be a collaborated attack among multiple attackers. The attackers do not need to trust each other to decide whether they have conducted the attack. Therefore, trust is not needed to collaborate since the attackers can always validate all information. The validation process is easy: if the number of outputs and inputs match, then the attack occurs.

In order to make the validation process faster, the attackers still need to share a part of the information related to the attacks they conduct. The information to be shared includes:

- A list of all outputs used in the attack.
- A list of all inputs involved in the attack.

4.8 Detecting the Attack

The strategy described in Section 4.3 is not applicable to mitigate the new attack; therefore, we define a new method. We assume that the shared information mentioned in Section 4.7 is kept secret among the attackers. We simulated this attack and determined two important features that distinguish malicious transactions and honest transactions:

- The malicious transactions repeatedly include a subset of outputs O as the mixins.
- A subset of the outputs O has a high usage value, which might be higher than the average usage value.

Precisely determining whether this type of attack has occurred in Monero and listing all related transactions might be infeasible due to the number of possible combinations. Consequently, we explore the features of this attack which will be identifiable within a set of transactions.

The diagrams in Figure 4.1 shows information regarding the average usage per output, the number of transactions and number of mixins aggregated for every 10,000 blocks. The information is useful to distinguish between regular output usages and suspected output usages. The sharp increase in diagram A is suspected to be caused by the mandatory RingCT scheme implementation, which left the users to have limited options of outputs to be used as their mixins. At the same time, the minimum ring size becomes five.

We then evaluated the data from the Monero blockchain up to block 1,542,882 containing 24.8 million outputs, where 4.7 million outputs are from RingCT transactions. Based on the finding, we divide the transactions into pre-RingCT and post-RingCT transactions due to their data characteristics. Figure 4.2 shows the related diagrams. The average output usage for all pre-RingCT transactions is 1.96, while the average output usage for all RingCT transactions is 3.68. Overall, the average output usage is 2.28.

The difference between pre-RingCT and post-RingCT transactions might also be affected by a change in the mixin sampling method. The triangular distribution replaced uniform distribution to increase the data resemblance with users behaviours as suggested by [73]. We simplify our analysis by ignoring the impact of the triangular distribution towards the evaluated data. We define an output weight of OW as the number of inputs where an output O perform as mixins in transactions. We also define an input weight IW as the average value of OW for all outputs in an input. IW is defined in Equation 4.1.

$$IW_s = \frac{\sum_{k=0}^{r} OW_k}{r} \tag{4.1}$$

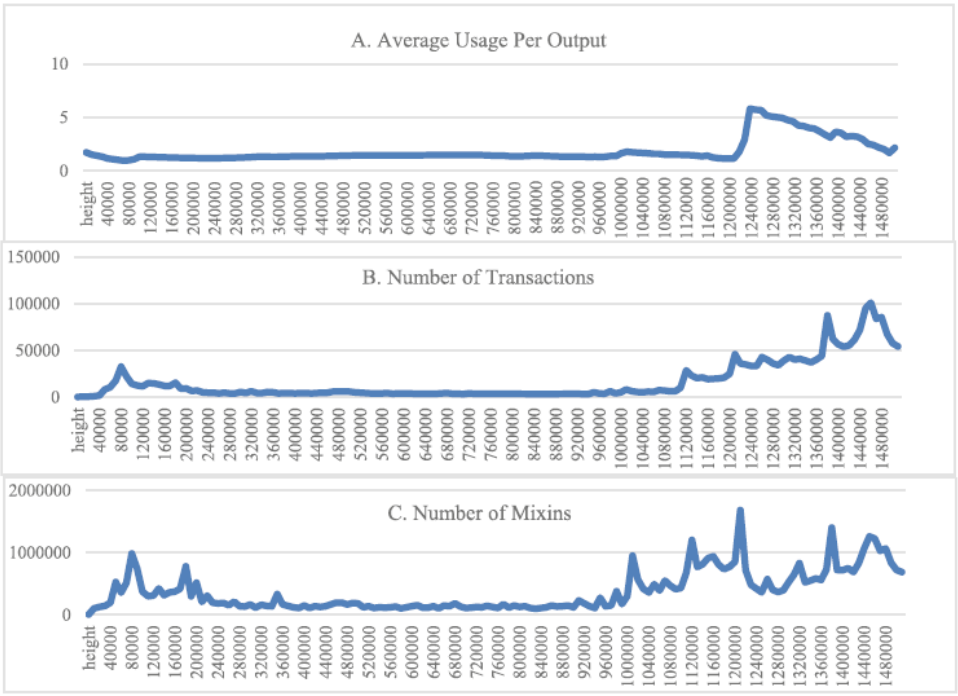

Figure 4.1: Diagram A shows the average usage per output from the blocks. Diagram B shows the number of transactions on the blocks. Diagram C shows the number of mixins of the transactions on the blocks. The horizontal axis shows the block height, while the vertical axis shows the value.

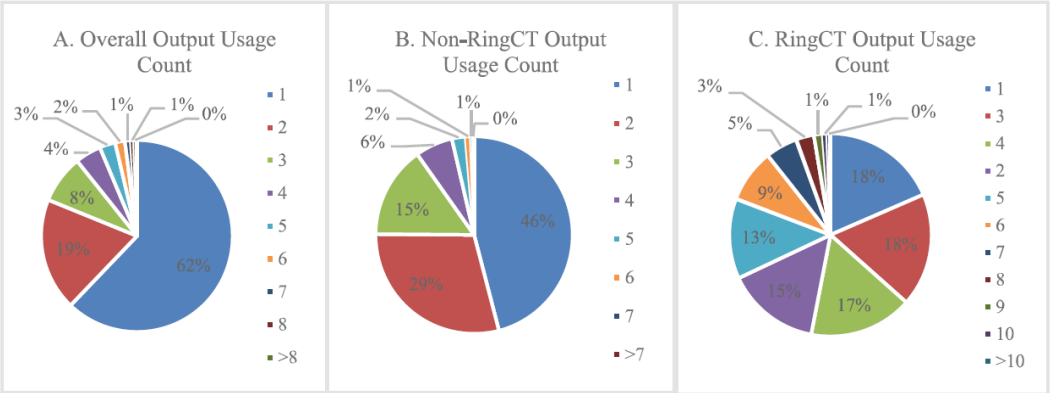

Figure 4.2: The aggregate data of all output usages with the legends describe the number of output usage. Diagram A shows the aggregate output usage count on Non-RingCT and RingCT transactions. Diagram B shows the same data on Non-RingCT transactions. Diagram C shows the data on RingCT transactions. We aggregate any data less than 1%.

We scanned the blockchain and computed the value of *IW* for all inputs up to block 1,545,153 (timestamped on 5 April 2018). We found a total of 45,650,192 inputs on the blockchain. The data is then aggregated and presented in Figure 4.3.

We look for standout figures on the blockchain data. We have presented the mechanism where IW is used to weight all mixins. We discovered that IW could perform as a unique value. Inputs that contain the same output having a high usage value are identifiable through IW. Based on our evaluation, inputs having IW value of at least seven are suspiciously reusing the same outputs multiple times. The total number of suspected inputs is 1,142,383 or 2% of all inputs recorded in the blockchain.

4.9 Mitigation Strategy: Input Weighting

The current triangular distribution sampling method does not guarantee that the users will use outputs that are not a part of anonymity reduction attack. Thus, there is an urgent need to improve the sampling protocol.

Our research shows that Monero outputs are not as fungible as claimed in [18]. Fungibility in Monero describes that all outputs have the same value regardless of who creates the outputs. Our results show that a subset of the outputs is potentially harming the anonymity of the transactions than other outputs. Hence, the term fungibility can also be applied to

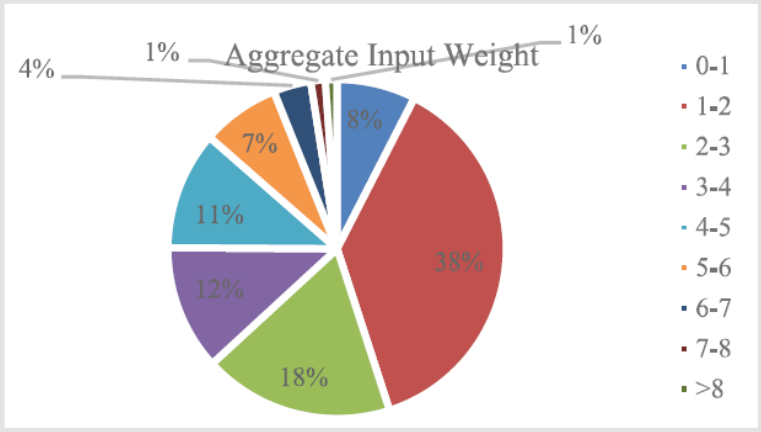

Figure 4.3: Aggregated IW for all transactions (nonRingCT and RingCT). The data is grouped based on the IW range. For example, the data marked as 1-2 means the IW is in the range of 1 to 2.

the mixins, since they determine the level of anonymity gained by the users.

To increase the anonymity and mitigate the attack, we propose the use of Input Weight (IW) as one of the criteria when sampling the outputs during mixin creation. The IW value of an output defines the probability of the output being a part of an attack.

Based on our evaluation, the current IW threshold to distinguish between normal transactions and suspicious transactions is seven. It is also possible that the threshold is changed due to changes in the system, primarily when the number of RingCT outputs increases or the mandatory ring size increases. The rule for determining the threshold is that the lower the threshold, the lower the risk would be.

4.10 Leveraging Monero UPID

4.10.1 Overview

The unencrypted Payment ID (UPID) poses an anonymity problem, where an observer can easily collect the information from the public blockchain and decode the message. A user investigated Monero payments associated with TheShadowBroker, a hacking group that wanted to auction their secret information gained unlawfully. The investigation managed

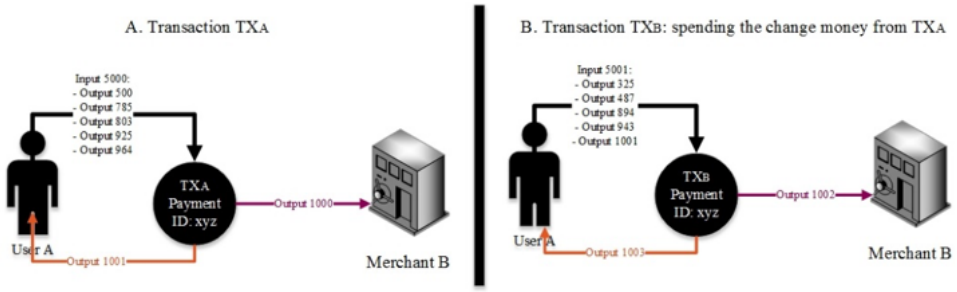

Figure 4.4: User A reusing the change money from the previous transaction in a new transaction. Both transactions are sent to the same merchant. Note `Output 1001` of TX_A from diagram A is included in `Input 5001` in TX_B.

to collect email addresses of TheShadowBrokers' clients[4].

By using a similar technique, we evaluated the use of UPID concerning the users' anonymity. The UPID is optional; hence, not every user utilises UPID. A user uses the same UPID to be included in multiple transactions when sending payments to the same merchant. Therefore, we assume that transactions with the same UPID are repeating payments with the same sender and receiver.

Based on the above scenario, if a user uses a UPID in a transaction and creates a second transaction including same UPID which includes outputs from the first transaction to the input mixins, then it is likely that the recent transaction spends these outputs. We consider the reused outputs are the change money. The change is not transferred to the receiver and returned to the sender's address. The scenario is described in Figure 4.4.

4.10.2 Results

We collected the transaction data from Monero blockchain and extracted the information into a relational database. The number of transactions using Payment ID is significant that more than half of the transactions ever recorded in the Monero blockchain are using Payment IDs, as shown in Figure 4.5.

From the genesis block up to block 1,535,607 (timestamped on 22 March 2018), we found 2,584,535 non-coinbase transactions (containing 23,108,911 inputs). With-in the

[4]https://steemit.com/shadowbrokers/@wh1sks/theshadowbrokers-may-have-received-up-to-1500-monero-usd66-000-from-their-june-monthly-dump-service

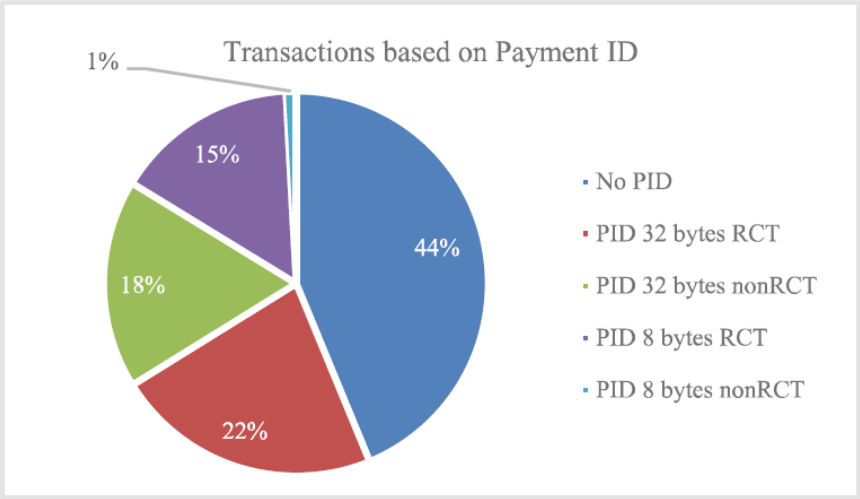

Figure 4.5: The transaction percentages based on Payment ID.

result, there are 1,033,891 transactions (containing 12,383,714 inputs) using UPIDs and 420,153 transactions using EPIDs.

We further investigated the transactions using UPID and managed to cluster the data based on the UPID reuse. There are 338,318 unique UPIDs found within the 1,033,891 transactions. There are also at least 15 UPIDs used more than 1,000 times. We then cross-referenced the transaction data to find identical UPIDs in multiple transactions. We discovered 332,987 traceable inputs from 165,919 different transactions using identical UPIDs. The identified inputs are 1.6% of total inputs in transactions using 32 bytes Payment ID.

We assume that the senders reusing the same UPID are sending money to the same merchants. We also assume that a part of the outputs is the change money transferred back to the senders' addresses. When the senders want to create other transactions to the same merchants, for example, to pay different purchases, then these senders can use the coins contained in the change addresses from previous transactions. Hence, we conclude that the new transactions are spending these outputs.

We investigated several cryptocurrency exchanges supporting Monero as one of their tradeable assets, such as HitBTC, Binance, Bitfinex, Poloniex, and Kraken. Based on information in Coinmarketcap.com, these cryptocurrency exchanges hold significant Monero trading volumes among other trading platforms.

Table 4.3 shows that three cryptocurrency exchanges are using UPID, namely Binance,

Table 4.3: A list of trading platforms and their Payment ID details. The information from cryptocurrency markets was collected on 4 April 2018. The trading volume data was taken from Coinmarketcap.com on 4 April 2018. The value of trading volume volume was calculated by summarizing all trading pair volumes. New deposit address/PID indicates that a user can create new deposit address or PID to deposit Monero on the cryptocurrency exchange.

No.	Platform	Trading Volume	Payment ID	New deposit address/PID
1	HitBTC	37.68%	EPID	No
2	Binance	16.67%	UPID	No
3	Bitfinex	13.84%	UPID	Yes
4	Poloniex	6.56%	EPID	No
5	Kraken	6.23%	EPID	Yes
6	Livecoin	3.82%	UPID	No

Bitfinex, and Poloniex. We can determine that Binance and Livecoin users will always have the same UPID for the same user, while Bitfinex is using the UPID but provides a feature where the users can regenerate the addresses and UPID by them-selves.

Repeated transactions are likely to be created by trading platforms or cryptocurrency exchange users. Research shows that the primary function of cryptocurrencies such as Bitcoin nowadays are tradeable assets rather than as a payment method [41]. Repeated deposits to the trading platforms are also possible, for example sending mining rewards directly from a mining pool to the miners' accounts in cryptocurrency markets.

Cryptocurrency trading platforms rely on the Payment ID to identify the customers' deposit. Without Payment ID, it is infeasible to distinguish the correct Monero transactions belonging to different customers. As their platforms may receive thousands of Monero deposits per day, the Payment ID is useful to automate the identification process, which will credit the correct customers' accounts with the correct amount of coins they transferred.

4.10.3 Possible Countermeasure: Encrypted Payment ID

We have presented a case where using the same UPID can be harmful to the users' anonymity, where an attacker can determine the real outputs spent by the trans-actions. The UPID is still widely used by cryptocurrency trading platforms. To mitigate the problem, users should abandon the UPID. The merchants also should modify their systems to support the

EPID, because EPID works identically to UPID. Hence there is no change in the merchants' business process that the correct accounts can be credited based on the payments received.

4.11 Conclusion and Future Work

In this research, we propose a mitigation strategy of an existing attack in [6]. Then, we formulate an extension of the attack, where the improvement of the new attack makes the previous mitigation method obsolete. By using distinguishable features we found in the transactions, we propose a simple approach yet effective as one of the considerations during the mixin sampling protocols. We also propose a second anonymity reduction attack by exploring the use of Payment ID. The Payment ID is a standard method being used by Monero merchants to distinguish payments from different users. We found that transactions having the same UPID is closely linked, such that at least 1.6% of the inputs are traceable.

For future works, we suggest the implementation of the proposed mitigation strategies in a working system. The hardened system contains all standard anonymity features such as traceable ring signature and one-time public key, including mitigation strategies as we have proposed in this chapter. Then, we will analyse the impact of the newly created wallet into the anonymity of the users and evaluate whether it is possible to construct new attack methods.

4.12 Chapter Summary

In this chapter, we improve our transaction-based attack, Monero Ring Attack (`MRA`) from the previous chapter, and propose Monero Ring Attack Extended (`MRAE`). Compared to `MRA`, `MRAE` is more difficult to detect, especially in a large-scale attack, because `MRAE` utilises unique combinations instead of `MRA`'s ring duplicates.

Both attacks demonstrate that transaction creation protocol is one of the most vulnerable protocols in Monero system. We propose to mitigate `MRA` by using a search-and-blacklist approach, while for `MRAE`, we propose new statistical metrics with an updateable threshold to identify the attack.

We conclude the first part of our project. Our next two chapters explore our second research question, `Q2`, that delves into Monero protocol updates.

Chapter 5

Anonymity Problems of Monero Protocol Updates

Monero ranked as one of the top privacy-preserving cryptocurrencies by market cap. The developers introduced semi-annual hard fork in 2018. Although the hard fork is not an uncommon event in the cryptocurrency industry, the two hard forks in 2018 caused an anonymity risk to Monero where transactions became traceable due to the problem of key reuse. This problem occurs due to the existence of multiple copies of the same coin on different Monero blockchain branches such that the users spent the coins multiple times without preemptive action.

In this chapter, we present the first part of our research on our second research question, `Q2`, about the risks of the asynchronous protocol update in Monero to the users' anonymity. We investigate the Monero hard fork events by analysing the transaction data on three different branches of the Monero blockchain. We also collect secondary data from cryptocurrency marketplaces for our analyses.

We discovered an insignificant portion of traceable inputs compared to the total available inputs in the blockchain. However, our analyses show that the scalability of the event depends on external factors such as market price and market availability. We propose a cheap, easy to implement a strategy to prevent the problem of key reuse, should in the future more durable Monero forks emerge in the market. Our published paper, *On The Unforkability of Monero*, is the primary resource for this chapter.

5.1 Introduction

Monero is one of the most successful privacy-preserving cryptocurrencies based on market cap[1]. Monero is a CryptoNote-based cryptocurrency [108]. Cryptographic methods such as Linkable Ring Signature (LRS) and one-time public key (OTPK) guarantee the untraceability of the sender and the unlinkability of the receiver. Transactions in CryptoNote protocol are more anonymous compared to other cryptocurrencies such as Bitcoin [108]. However, research shows that the transactions in Monero can be traceable due to the problem of zero mixin transactions [59, 79].

In a blockchain ecosystem, a fork is an event where the protocol changes [126]. Zamyatin et al. [126] classified the changes into several possibilities: protocol expansion, reduction, conflicting (bilateral), and conditionally reduction (velvet). There are generally two types of blockchain fork, namely hard fork and soft fork. A hard fork is related to either protocol expansion or bilateral, which results in a blockchain split or chain split. A soft fork, including protocol reduction and velvet, does not produce any chain split [126].

Blockchain fork in the cryptocurrency setting usually benefits the users financially [93]. For example, a user had one Bitcoin before the Bitcoin Cash fork. After the forking event happened, the same user will double his/her money: one Bitcoin in the original blockchain and one Bitcoin Cash in the newly created blockchain branch. The price of Bitcoin was US$2,808 (open price)[2] while Bitcoin Cash was traded at the rate US$555.89 (open price) each[3], so the user received a coin worth US$555.89 for every bitcoin he/she held by doing nothing.

As a part of the attempt to keep its system updated, Monero has scheduled semi-annual hard fork. The hard fork of Monero mostly aims to improve privacy solutions. However, Monero also intends to become an ASIC-proof cryptocurrency, where the software developers discourage the development of ASIC machines for participating in Monero mining through the existing Proof-of-Work (PoW) mechanism [108, 30]. For this purpose, on 6 April 2018 Monero updated its mining algorithm to render existing ASIC machines' efficient computing void. The event created a new Monero fork as Monero upgraded from protocol version six to version seven.

The hard fork happened due to the incompatibility between two protocol versions. The protocol version seven started from block number 1,546,000[4], which was on 6 April

[1]The total market cap for Monero is US$1.7B according to Coinmarketcap.com on 26 October 2018
[2]https://coinmarketcap.com/currencies/bitcoin/historical-data/?start=20170723&end=20170723
[3]https://coinmarketcap.com/currencies/bitcoin-cash/historical-data/?start=20170723&end=20170723
[4]https://github.com/monero-project/monero

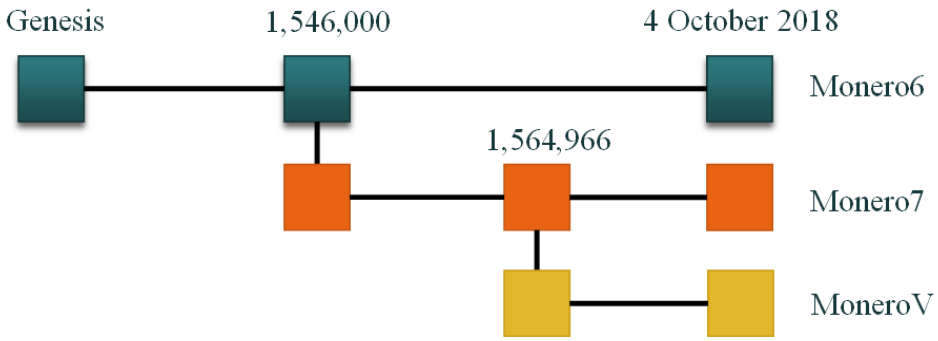

Figure 5.1: The Monero hard fork timeline which shows two hard forks resulting in three different chains by October 2018.

2018. Several independent parties were claiming that they still wanted to run version six protocol. These independent parties rebranded the blockchain system running Monero protocol version six into three different names: Monero Original (XMO), Monero 0 (Monero Zero or ZMR), and Monero Classic (XMC)[74]. Although there have three different names, in reality, they only have one blockchain [7]. After the Monero hard fork, another project was also forked from the Monero version seven, called MoneroV (XMV), starting from block number 1,564,966[5] which was on 3 May 2018. MoneroV runs a modified version of protocol seven. One of the modifications on MoneroV's protocol from Monero version seven is the reduction of the decimal places from 12 to 11. Hence, the value of all coins multiplied tenfold. The fork history is as in Figure 5.1.

Similar to other blockchain fork events, the Monero users doubled their Monero coins on each event, where they receive the same amount or even more coins in the newly created cryptocurrencies. However, when the users decide to use the same coins without extra caution, the anonymity of the transactions is potentially reduced.

A Monero hard fork event can compromise transaction anonymity. The hard fork event creates new coins spendable by the pre-fork users. The problem occurs when the Monero users spend the newly created coins. Hence, the traceability of the transaction's sender is deducible. We denote this as the problem of key reuse. The problem of key reuse reduces the anonymity of a transaction because the users reuse the same keys when spending the same coins. Monero developers acknowledged the problem of key reuse [30]. However, the existing strategies are not sufficient to mitigate the problem completely.

Contributions. We summarise our contributions as follows.

[5]https://github.com/monerov/monerov

- We investigate the impact of two Monero hard forks which occurred in 2018. We collected the data of each blockchain branch and cross-referenced the transaction data on all of the blockchain branches to determine traceable inputs. We discover over `55K` distinct traceable inputs, where 90% of them are from two blockchain branches running version six and seven. About 19% of all inputs in Monero version six of our dataset are traceable. However, the percentage of the traceable inputs is insignificant compared to the total inputs in Monero version seven (or the main branch). Unlike existing attacks [79, 116, 125, 59, 117], the analysis on the problem of key reuse can identify traceable inputs in Monero's RingCT. Existing attacks cannot discover any of the traceable inputs in our findings.

- We analyse the correlation between the identified traceable inputs and market prices by using statistical analyses, namely correlation coefficients and linear regression. The results on the correlation coefficients show that there is a strong correlation (0.553 to 0.761) between the two factors in Monero6, whereas we discovered a weak correlation (0.242 to 0.32) in MoneroV. Likewise, the linear regression results show a medium relationship in Monero6 but a small relationship in MoneroV. More significant support from cryptocurrency exchanges to Monero6 compared to the support to MoneroV also explains the massive gap of traceable inputs on both blockchain branches. These findings also show that the scalability of the problem of key reuse depends on the identified external factors.

- We propose a mitigation strategy for the problem of key reuse caused by hard forks in Monero. `Scalable Bloom Filters` [1] provide an inexpensive checking mechanism of existing key images and mixins. By adding the checking mechanism before submitting new transactions to the network, transactions spending the same coins will maintain the same level of anonymity. We also propose a new service node called `joint node` as a part of the mitigation strategy. The node protocol is compatible with the current Monero protocol without any significant changes.

Organisation. The organisation of this chapter is as follows. Section 5.2 contains related work about existing anonymity attacks to Monero, an alternative strategy to a hard fork, and replay protection. A threat model is available in Section 5.3. Section 5.4 describes our analysis methods. Section 5.5 covers current mitigation strategies to the identified problems. Our mitigation strategy proposal is also in this section. Section 5.6 provides discussions on the security and the performance of the proposed mitigation strategy, while Section 5.7 concludes the findings and describes future work.

5.2 Related Work

5.2.1 Velvet Fork

Velvet fork is a term coined by Kiayias et al. [57]. Rather than having a hard fork, the velvet fork offers a new strategy where variables replace constant parameters when changing the protocol. By using velvet fork, the hard fork is avoidable [126].

Although velvet fork is useful to avoid a hard fork that changes parameters, the velvet fork is unusable when the change is in the protocol layer. If Monero adopts velvet fork technique, then increasing ring size would be effortless by modifying a prepared variable holding the ring size value. However, protocol layer changes such as modifying decoy selection method, adding new signature feature [100], or changing consensus method [30] cannot be done by using velvet fork.

5.2.2 Replay Protection

Replay protection is a mechanism to avoid a replay attack. A replay attack in cryptocurrency refers to retransmission of a valid transaction data on a system to other compatible systems [71]. The result of the replay attack is that the payee will get multiple payments in different cryptocurrencies such that the payer suffers a loss. The paper by McCorry et al. [71] describes several examples of replay protections being implemented in different cryptocurrencies, such as `Chain ID`, `Transaction Version`, `Check Block At Height`, and `Sighash Enum`.

`Chain ID` has been implemented in Ethereum since `Spurious Dragon` hard fork [20], while `Transaction Version`, `Check Block At Height`, and `Sighash Enum` were proposed as alternative solutions for replay protection in Bitcoin [71]. A new proposal introduced new replay protection methods, namely `Migration Input` and `Hardfork Oracle` [71]. `Migration Input` was proposed to be implemented in Bitcoin protocol, where the input hash is modified from 32 bytes to 41 bytes to accommodate extra information. By having a different input hash scheme, the old protocol will not be able to validate the transaction, and therefore only the new compatible protocol validates the transaction [71]. `Hardfork Oracle` was proposed to be implemented as an Ethereum smart contract as an automatic detection tool of transactions from different forks.

There is no replay protection currently being implemented in Monero protocol [90]. One of the ways to create a transaction with built-in replay protection is to include at least one new output in the intended chain as one of the decoys in the ring signature [105].

5.2.3 Attacks on Monero Protocol Update

The asynchronous protocol update or hard fork, on Monero can also lead to attacks [118]. A research discovered that the nodes that have not yet updated their applications to the latest version (which run the latest protocol version) are prone to Denial of Service (DoS) attack, where a large number of transactions can potentially flood the nodes' temporary storage *txpool*.

Furthermore, the DoS attack can announce the traceability of the inputs to the public by submitting two different transactions that spend the same coins: one transaction submitted to the old nodes (which run the old protocol) and another to the new nodes (which run the new protocol). If the required condition of the network holds, then new transactions in the old nodes will never be confirmed to the network. Hence the double spending will never occur. However, since two coins appear twice in different nodes (which run different protocols), then the real inputs of the related transactions can be deduced.

5.3 Threat Model

The untraceability in Monero requires that an observer cannot guess the real output being spent in a ring construction R with a probability of more than $\frac{1}{r}$, where r is the number of ring members. In this case, the anonymity of the real output depends on the size of r.

We define an input I_j traceable to an output o_j as follows. The output o_j is an output of a blockchain B_1 in a linkable ring signature R_1 with a set of output $O_1 = \{o_a, \ldots, o_j, \ldots, o_l\}$ as its ring members and key image k_j, such that R_1, k_j, and O_1 are parts of I_j. The same output o_j appears on another blockchain B_2 which is included in another linkable ring signature R_2 with a set of output $O_2 = \{o_m, \ldots, o_j, \ldots, o_z\}$ and key image k_j. From this occurrence, it can be concluded that the key image k_j is associated to the output o_j. Therefore, the input I_j is traceable to the output o_j, because the probability of guessing the real output is 1. This occurrence also fulfills the linkable condition as described in Linkable Ring Signature [64] as it can be concluded that the two ring signatures R_1 and R_2 are created by the same person, assuming that the secret key image k_j is only known to the owner.

Anonymity reduction occurs when q members of a ring R are deducible since they are no longer fit as the candidate when guessing the real output. Therefore the anonymity r is reduced by q. We define a reduced anonymity input I_i as an input of a blockchain B_1 in a linkable ring signature R_3 with r as the ring size using a set of

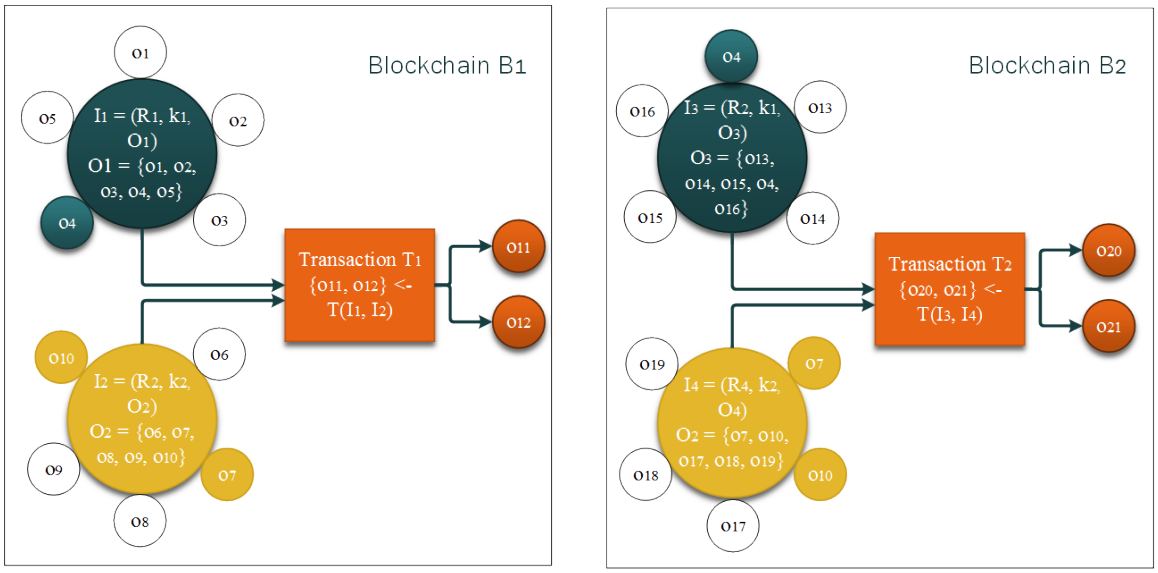

Figure 5.2: The inputs I_1 and I_3 have two common similarities: the key image k_1 and the output o_4. Both inputs I_1 and I_3 are traceable. The inputs I_2 and I_4 contains the same key image k_2, however there are two identical outputs in the ring, namely o_7 and o_{10}. Both inputs I_2 and I_4 suffer anonymity reduction by three.

output $O_3 = \{o_b, \ldots, o_e, \ldots, o_i, \ldots, o_m\}$ and key image k_h. The same key image k_h was found on a ring signature R_4 recorded on another blockchain B_2 using a set of output $O_4 = \{o_c, \ldots, o_e, \ldots, o_i, \ldots, o_n\}$ where at least two outputs $\{o_e, o_i\}$ in R_4 fulfill the following criteria: $\{o_e, o_i\} \in O_4$ and $\{o_e, o_i\} \in O_3$. In this scenario, it is inconclusive whether key image k_h is associated to o_e or o_i. The example of a traceable output and an anonymity reduction is shown in Figure 5.2.

We define passive attack and active attack in Monero. In the passive attack, an attacker collects information from the public blockchain(s) and conduct traceability analyses. In the active attack, the attacker controls dishonest nodes which return false information. By returning false responses of key image-related requests, the dishonest nodes expect that the client suffers anonymity reduction from the problem of key reuse, especially when a client spends the same coins in different blockchains.

We assume that all cryptocurrency backers, including software developers, community members, and users, desire the best privacy-preserving features for their systems. However, there also exist some users in the system who unknowingly spend identical coins in multiple blockchains. Therefore, related transactions become traceable. The events cause cascade

effect, which make other transactions traceable or suffer anonymity reduction [79]. We assume that the majority of the system nodes behave honestly by sending the correct responses or information from any requests.

We assume that the current system can only accept small modifications which do not substantially affect how the whole protocol runs. However, we also assume that hard forks can occur at any given time such that the same unspent coins before the fork can be spent multiple times on different blockchains after the fork. Conducting a hard fork is incentivised financially as the newly created coins can be sold in the markets.

5.4 Analyses

5.4.1 Analysis on Traceable Inputs

We collected all transaction data on three different blockchains, which we called Monero6, Monero7, and MoneroV. We use the term Monero6 to refer cryptocurrencies that are using the Monero protocol version six: Monero Original, Monero 0, and Monero Classic. Monero7 is used to refer the Monero main blockchain which runs protocol version seven (as of October 2018). MoneroV is self-explanatory, referring to the blockchain system which runs MoneroV protocol.

A checking algorithm is as follows.

1. We initialise an array K_{res}. The array will store the identified key images as the final result.
2. For each key image, $k_i \in K$ do the following steps.
   (a) Compute the total number of its occurrence in all three blockchains and store the result in a variable, *occ*.
   (b) Compute the number of unique transaction hash ($txhash$) and store the result in another variable, *unq*.
   (c) Execute a conditional statement as follows: if $(occ > 1)$ and $(unq > 1)$ then $K_{res} \leftarrow k_i$. The conditional statement is required to filter out key replay cases as they do not help to identify the real output.
3. Return K_{res}.

Table 5.1: The summary of traceable inputs from the problem of key reuse and the cascade effect it caused.

Blockchain	Height	Key Reuse		Cascade Effect		Dataset	
		Traceable Input	Tx.Count	Traceable Input	Tx.Count	Input Count	Tx.Count
Monero6	1,675,606	52,646	3,148	278	276	274,131	44,467
Monero7	1,675,303	53,162	5,680	315	312	1,876,341	810,409
MoneroV	1,671,617	7,542	888	0	0	269,335	84,053

We have examined three blockchains by using the algorithm above on a dataset we built by extracting non-coinbase transactions (transactions that are not block reward) confirmed on block number 1,546,000 to 1,675,606 in *Monero*6 (181 days period), block number 1,546,000 to 1,675,303 in *Monero*7 (181 days period), and block number 1,564,966 to 1,671,617 in *MoneroV* (152 days period). The cut-off period for data extraction is 4 October 2018. We also have conducted a cascade effect analysis to determine how many traceable inputs as the impact of the problem of key reuse on the same dataset. The result is presented in the Table 5.1.

Based on the algorithm we developed, we discovered 52,924 traceable inputs on Monero6 (including the ones discovered using cascade effect method), 53,477 traceable inputs on Monero7. In contrast, there are only 7,542 traceable inputs found on MoneroV. Although there is only 29 days difference between MoneroV and both Monero6 and Monero7, the traceable inputs found on MoneroV is only around 14% of traceable inputs found on both Monero6 and Monero7. There is also an extreme difference in the number of non-coinbase transactions between Monero7 and the other two cryptocurrencies. Monero7 has 810,409 transactions, where Monero6 and MoneroV only contain 5% and 10% of Monero7's transactions, respectively.

The traceable inputs in Monero6 are 19% of Monero6's total number of inputs in our dataset, whereas the traceable inputs in Monero7 and MoneroV are only 2% of their total inputs. The result shows that the problem of key reuse has a more significant impact on Monero6 than on Monero7 and MoneroV. Also, about 90% of all traceable inputs are found among Monero6 and Monero7, whereas only 6% of the traceable inputs are on all three blockchain branches. The numbers show that Monero6 is the higher source of the problem of key reuse to Monero7 than MoneroV.

5.4.2 Analysis on Anonymity Reduction

We also examined reduced anonymity as the side effect of the traceable inputs and cascade effect. An input with a ring size of r suffers anonymity reduction if there are at most $r-2$ spent ring members. By suffering reduced anonymity, the real output is still untraceable. However, the probability of guessing the real output increases from $\frac{1}{r}$ to $\frac{1}{r-u}$, where $1 \leq u \leq (r-2)$ and u is the number of spent outputs.

We discovered 1,848 inputs in Monero6 that suffer anonymity reduction. Likewise, 2,819 inputs in Monero7 and 264 ring signatures in MoneroV suffer anonymity reduction while still being untraceable. Figure 5.3 shows the trend of the anonymity reduction on all three blockchain branches. About 95% of the anonymity reductions have a reduced size of u between 1 to 5.

The result shows that a transaction having a ring size of 5 or less is riskier than one having a ring size of more than 5. We calculated the average ring size for the three blockchain branches starting from the first block of the fork to the cut-off period on 4 October 2018, where Monero6, Monero7, and MoneroV have an average ring size r of 5.07, 7.56, and 7.75 respectively. Having a bigger ring size provides better protection on the anonymity of the input. However, creating a transaction with a bigger ring size may result in a more expensive transaction fee to be paid by a user. Determining a bigger minimum ring size in the protocol level can also be helpful to ensure that the users have sufficient anonymity in their transactions. While Monero6's minimum ring size was five, Monero7 and MoneroV have a minimum ring size of seven. Based on the result, it is advisable to have a ring size $r > 5$.

5.4.3 Analysis on Key Reuse and Cryptocurrency Market Price Correlation

We collected historical prices of Monero Classic and Monero Original from Coinmarketcap.com (20 April 2018 to 3 October 2018), while the prices of MoneroV were from Coingecko.com (4 July 2018 to 3 October 2018). We calculated statistical data regarding the market prices for Monero Classic, Monero Original, and MoneroV. The summary of the market prices is on Table 5.2. We identified that while the prices of Monero Classic and Monero Original are in the average of US$4 (lowest at US$1 and highest at US$27), the average price of MoneroV is meagre, which is at US$ 0.06 (lowest at US$0.02 and highest at US$0.26). If the price of MoneroV is multiplied by ten, then the MoneroV price only

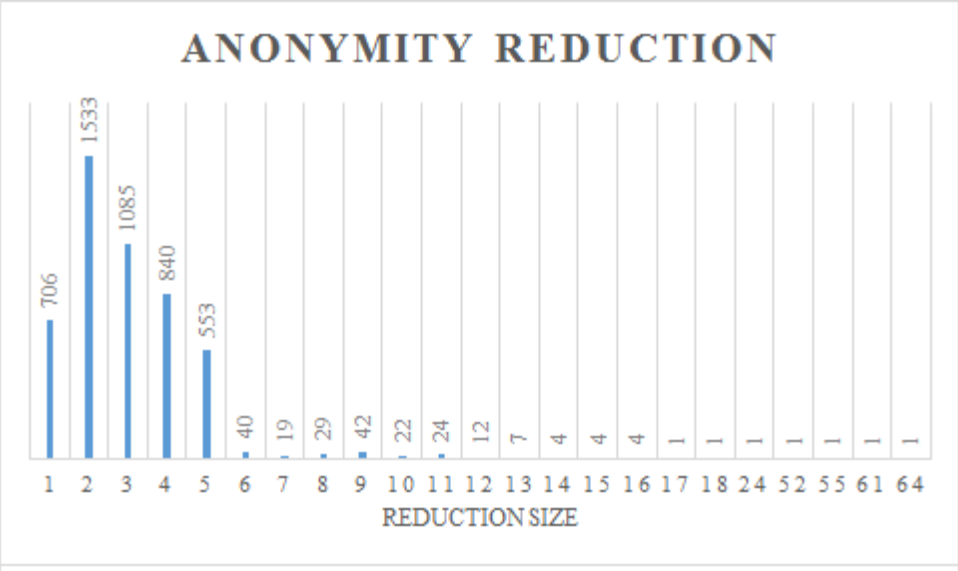

Figure 5.3: The summary of anonymity reduction as the result of key reuse problem on Monero6, Monero7, and MoneroV.

becomes US$0.6 on average. In contrast, the average price of Monero Classic and Monero Original is seven times more expensive.

Based on the price information of Monero Classic (price at high, low, and close), Monero Original (price at high, low, close), and MoneroV (price at open and close), we studied the correlation between market prices and traceable input count as well as transaction count by computing **Kendall's tau-b correlation coefficient** and **Spearman's rho correlation coefficient**. The result is presented in Table 5.3.

Cohen as cited by Sauro and Lewis [97] provided three interpretations of correlation coefficient r as follows: r is small when $0.1 \leq r \leq 0.3$, medium when $0.3 \leq r \leq 0.5$, and large when $r \geq 0.5$. Our results show that all coefficients for both Monero Classic and Monero Original are in the range of 0.557 and 0.754, which indicates a strong correlation between the price and key reuse in Monero Classic and Monero Original. On the other hand, the result shows a small correlation between the price and key reuse in MoneroV, where the coefficients are 0.242 and 0.32. Based on this information, we deduct that the market price and the number of traceable inputs are correlated. However, the statistical analyses cannot determine the causality between the two parameters.

Figure 5.4 shows the linear regression analysis of the dataset. The R-squared values are 0.180, 0.146, and 0.003 for Monero Classic, Monero Original, and MoneroV, respectively.

Table 5.2: The summary of market prices of Monero Classic, Monero Original, and MoneroV.

	Monero Classic				Monero Original				MoneroV	
	Open	High	Low	Close	Open	High	Low	Close	Open	Close
Max.Price	21.99	27.42	18.02	21.55	21.81	24.36	14.88	20.75	0.26	0.26
Min.Price	1.20	1.29	1.04	1.20	1.25	1.29	1.22	1.23	0.02	0.02
Avg.Price	4.38	4.84	4.04	4.34	4.24	4.64	3.92	4.21	0.06	0.06

Table 5.3: Correlation between the number of traceable input and market price (open price) of Monero Classic, Monero Original, and MoneroV

	Correlation	
	Kendall's Tau-b	Spearman's Rho
Monero Classic	0.561	0.755
Monero Original	0.557	0.754
MoneroV	0.242	0.32

The result shows a medium relationship between the number of traceable inputs and the market price on both Monero Classic and Monero Original, but a small relationship among the two variables on MoneroV.

Figure 5.4: The linear regression of the number of traceable inputs and market price of Monero Classic (figure a), Monero Original (figure b), and MoneroV (figure c).

5.4.4 Analysis on Key Reuse and Coin Availability

At the time of writing, cryptocurrency portal Coinmarketcap.com lists Monero Classic (XMC) and Monero Original (XMO) as two separate coins. In reality, these two cryptocurrencies are in the same blockchain branch. Each of these cryptocurrencies is exchangeable different cryptocurrency markets. Monero Classic[6] is traded at:

- Gate.io (XMC/USDT pair and XMC/BTC pair)
- HitBTC (XMC/BTC pair, XMC/USDT pair, and XMC/ETH pair)
- TradeOgre (XMC/BTC pair)

Monero Original, on the other hand, is only available at HitBTC (XMO/BTC pair, XMO/USDT pair, and XMO/ETH pair). The other Monero fork, MoneroV (XMV), is only available at TradeOgre (XMV/BTC pair). On 4 April 2018, two days before Monero6 fork, the top six most busy cryptocurrency exchanges serving Monero are HitBTC, Binance, Bitfinex, Poloniex, Kraken, and Livecoin, which made 84.8% of the total daily Monero trading volume [117]. HitBTC has the largest portion of 37.68%, while Livecoin had the lowest portion of 3.82% among the six exchangers [117]. Although the Monero pre-fork trading information history in TradeOgre is unavailable, it still implies that the majority of the cryptocurrency markets did not support MoneroV. Therefore the Monero users never received MoneroV from the cryptocurrency exchanges. On the other hand, as Monero6 received support from big exchanges such as HitBTC, the number of Monero6 coins distributed to the users were much larger than MoneroV coins.

The market availability is likely to be one of the factors that cause the number of traceable inputs on MoneroV is only 14.3% of Monero6's traceable inputs. However, it is not possible to determine how many spent coins in the traceable inputs due to the implementation of `RingCT`, which obfuscates the number of coins involved in the transactions.

We also made several assumptions as follows.

1. The users will be given free coins after a hard fork, as long as they hold some coins on the original blockchain branch before the hard fork [51, 49, 50].
2. The users prefer to get a maximum benefit by selling the coins at a high price.

[6]Information about Monero Classic and Monero Original is taken from Coinmarketcap.com, while information about MoneroV is from Coingecko.com.

3. If the users are using private wallets (either by using a computer wallet, smartphone wallet, web wallet, or paper wallet), then the users need to set up new wallet applications for the new chains and import the information from their old wallets. Then, they can create transactions.

4. If the users deposit their coins to cryptocurrency exchanges' wallets before the hard fork, then it depends on the exchanges whether they are supporting the new coins. Then the users will receive the new coins from the exchanges in their accounts on those exchanges [51, 49, 50].

The assumption 1 and 2 motivate the users to redeem their new coins. Hence, the problem of key reuse potentially occurs. However, assumption 3 becomes a barrier for the users, as setting up new wallets by importing private keys from old wallets is not trivial and requires technical knowledge. In assumption 4, if the cryptocurrency exchanges do not support the forked coins, the users will not receive any new coins.

5.5 Mitigation Strategies

In this section, we discuss existing mitigation strategies for the problem of key reuse. We also propose a new mitigation strategy.

5.5.1 Current Mitigation Strategy

5.5.1.1 Not claiming the coins

There is a suggestion for Monero users not to use the newly created coins they received for free [98]. The number of coins received by the users varies depending on the new systems. Monero6 provides 1:1 coin distribution, where each Monero holder receives the exact amount of coins on the new blockchain branch. MoneroV offers 1:10 coin distribution, in which the Monero holders receive ten times the amount of Monero they have before the fork occurs.

There are two identified problems in this method. Firstly, the method is not preferable by the users, since they may lose potential profits by not claiming coins. The free coins are tradeable for other cryptocurrencies or even local currencies. The suggestion is even more unlikely to be followed by the users when the amount of profit they can get by spending the coins is substantial. Secondly, the method can only be effective if and only if all Monero users do not redeem their new coins. If any of the Monero users redeem the new coins, the

redeeming transactions can potentially reduce or even altogether remove the anonymity of other users' transactions which make their transactions traceable.

5.5.1.2 Churning

Churning technique expands the output selection by sending the users' coin to their addresses multiple times [58]. By churning the coins, the number of ring members or decoys will expand, which in turn makes it harder for an attacker to determine the real outputs from multiple transactions created during the churning process to trace the transactions. However, the churning technique is still prone to known attacks such as timing attack and network attack, which deduct real outputs based on the origins of the transactions [98]. The effectiveness of churning to mitigate the key reuse problem is also questionable [98].

5.5.1.3 Blackball tool

The Monero developers created a tool called `blackball`. `Blackball` is a term originally coming from a document that describes the first identified problem within Monero system, where an adversary tries to add his/her outputs to the blockchain by creating as many transactions as possible [85]. The outputs controlled by the adversary is called `blackballs` or `black marbles`, while other outputs created by genuine users are `white balls` or `white marbles`. Whenever the blackballs are included in a transaction t as members of the mixins, then the adversary will be able to determine his/her `blackballs` as decoys and not the real outputs. Hence the anonymity level of the affected transaction t is reduced.

The `blackball` application created by the Monero developers tries to blacklist known malicious outputs; all outputs in the blacklist are discouraged from being used by the users as mixins. The application is not mandatory. The official Monero node application and Monero wallet did not have the blackball feature.

The problem with the `blackball` tool is that the users need to have a full copy of each blockchain branch, which then the tool will compare and extract the information from the blockchain branches. However, assuming one blockchain branch requires $50GB$ of storage space, then three blockchain branches will require at least $150GB$ of storage space, just to keep their anonymity level intact. The `blackball` cannot run on light hardware such as smartphones, where space and computing power is limited. Hence it is unlikely that all users can run `blackball` tool to make their transactions safe. Other than the `blackball` tool as mentioned earlier, there are features in the default/official `CLI`-based Monero wallet which can reduce the problem of key reuse.

- To let users set the mixins/decoys themselves manually. This feature is implemented in `set_ring` command [106].
- To only use mixins that exist before the fork. This feature is implemented in `segregate-pre-fork-outputs` command [106].
- To combine mixins from before fork and after the fork. This feature is implemented in `key-reuse-mitigation2` command [106].

Having identical mixins in transactions in multiple blockchain systems will prevent the passive attacker from tracing the transactions because the anonymity level is not compromised. However, there is no satisfactory solution to help the users conducting the best practice in maintaining the anonymity of their transactions.

5.5.2 Our Proposed Solution

Our proposed solution consists of three parts, namely hard fork management, key image management, and joint nodes.

5.5.2.1 Hard Fork Management

We propose to add `Chain_ID` information in every transaction, which will be useful for several reasons. Firstly, the `Chain_ID` prevents a replay attack. Secondly, the `Chain_ID` poses as an identifier when the users want to get information about outputs (when they want to create new transactions) or existing key images.

To complement the `Chain_ID`, a `Fork_Point` information also needs to be managed. `Fork Point` is the first block height of a new chain that has a different block hash compared to its parent, which is similar to Ethereum's `FORK_BLKNUM` [20]. Unlike `Chain_ID`, the `Fork_Point` does not need to be embedded to transaction data. The reason for not embedding the `Fork_Point` in the block or in the transaction is to save space from less useful information. For this requirement, a new database called `Chain_Info` will be created. The new database contains both `Chain_ID` (as the primary key) and `Fork_Point`.

Chain_Info. The proposed solution must record new chains after hard forks in a `First-Come-First-Serve` basis. The `Chain_ID` can be used to query the `Fork Point` from the newly created `Chain_Info` database. The `Chain_Info` database is inside the node's blockchain database file. This approach is different compared to Ethereum's method which

stores the `Chain_ID` information on a Github page [20], while Monero stored the history information of its own hard fork by *hardcoding* it to the source code[7]. However, this approach will be infeasible when dealing with external hard forks, where the occurrences might not be known to the Monero developers and Monero community.

5.5.2.2 Key Image Management

We have identified several issues about key image information management, including:

1. Multiple blockchain branches having different block interval.
2. Multiple transactions submission having identical key images in a short period of times
3. Updatability of the key image ring members (e.g. member addition).
4. New transactions with ring members that are not available in the parent chain where the key images first appeared.
5. New chains that have smaller mandatory ring sizes compared to the parent chain.

Scalable Bloom Filter. Scalable Bloom Filters [1] can solve the identified issues. Scalable Bloom Filters (`SBF`) is an extended version of the original Bloom Filter [14] where scaling is enabled by utilising multiple Bloom Filters instead of a single filter as in the standard Bloom Filter (`BF`). Therefore, the capacity in `SBF` can be expanded after initialisation, contrary to BF, which cannot exceed the predefined capacity. Similar to BF, an `SBF` can produce false-positive results where the filter detects that a data is in a setting where it should not. However, `SBF` also inherits the characteristics of BF, where false negative is negligible. A false negative is when the BF returns `False` (that the data is not in the set) when it is supposed to be `True` (the data is actually in the set).

In our proposed solution, we introduce several `SBF`s. The first `SBF` is used to filter key images, namely SBF_k. Key images from related blockchains (parents, siblings, or child chains) are inputs to compute SBF_k. By constructing SBF_k, new key images are identifiable whether they have existed in any blockchains, such that when the checking algorithm result is true, then the protocol raises a flag to avoid the problem of key reuse. The second SBF filters hash values of key image-mixin tuples, namely SBF_m. Similar to

[7] https://github.com/monero-project/monero/blob/master/src/cryptonote_core /blockchain.cpp#L120

SBF_k, SBF_m consists of key image-mixin tuples from all related blockchains. The purpose of SBF_m is to help the system to identify whether an incoming key image-mixin tuple has existed in one or more blockchains.

Despite its scalability feature, SBF does not support data deletion. Therefore, to mitigate different block intervals and block reorganisation where other stronger blocks can still replace immature blocks, temporary $SBFs$ are introduced. These temporary $SBFs$, namely $tSBF_k$ and $tSBF_m$, are associated to key images and key image-mixin tuples respectively. By using temporary $SBFs$, it means new information in immature blocks and memory pools will not go directly to the main $SBFs$ but to $tSBFs$. After the information is available in mature blocks, the protocol moves the data to the main $SBFs$. SBF_k and SBF_m can complement each other by the following mechanism.

1. The system checks a key image value in SBF_k. If it does not exist, then checking process complete, otherwise continue. In this step, it not possible to know whether the checking result is a false positive result or a true positive result.

2. We define t as the threshold to be satisfied by new transactions.

3. For each ring signature R with a ring size r, there will be r key image-mixin tuples. The system checks every key image-mixin tuple if they exist in SBF_m and count the positive results p. If $p = r$, then there is a possibility (due to `SBF`'s false positive characteristic) that the input has the same ring members as the existing input, and this is a desirable occurrence. However, that might not always be the case. It is possible that $p < r$, but as long as p can satisfy the threshold t where $t > 1$, then $t <= p <= r$ must be satisfied. Otherwise:

   (a) If $t = 0$ then the transaction that contains the ring signature R can be accepted as it is a false positive caused by SBF_k.

   (b) If $t = 1$ then the transaction that contains the ring signature R can be blacklisted as it can cause traceable output.

4. To increase the probability of the new transactions using an identical set of the existing ring members, the threshold t can be set to $t = r$ such that $t = p = r$.

Blacklisting can be an option instead of rejecting the transaction, as described in the step 3b because transaction rejection might motivate users to recreate the transaction, which will make the new transaction traceable [76].

False Positive. The false-positive rate is the trade-off for not using the real transaction data in our solution, which focuses on cost-efficient by using SBFs. The error rate in the original design of SBF is expected to be between 0.0001% to 0.1% [1]. The use of two different $SBFs$, namely SBF_k and SBF_m greatly reduce the false positive rate in the case of a new key image that never appears, such that the false-positive result indicates otherwise.

We utilise a simple equation of probability of two independent events $P = p_1 \times p_2$ where p_1 and p_2 are the probabilities of the first and the second event, respectively. By using the equation and taking the most significant error rate of SBF, the probability of a key image that never appears before as false positive in both checks is 0.0001%.

SBF for Multiple Blockchain Branches. An `SBF` consists of multiple Bloom Filters (BF) [1] where $SBF = \{BF_1||BF_2||\dots\}$, where the symbol $||$ is a concatenation operation. An `SBF` can also be constructed by concatenating multiple `SBF`s such that $SBF_{result} = \{SBF_a||SBF_b||\dots\}$. We denote `Local SBF`s (`LSBF`s) as a set of `SBF`s which are created by using information from a single blockchain branch. We also denote `Global SBFs` (`GSBF`s) as a set of `SBFs` which are created by concatenating all `Local SBFs`. The `GSBF`s are used to check the existence of a related information regardless in which blockchain the information resides, while the `LSBF`s are used to check information on a specific blockchain.

SBFChain. For accountability purposes of the created `GSBF`s, we introduce `SBFChain`. `SBFChain` is a blockchain-like data structure which maintains metadata about the `GSBF`s and tracks changes to the `GSBF`s. Each entry in `SBFChain` is numbered. An entry e_n in the `SBFChain` connects to the entry e_{n-1} by adding the hash value $h_{e_{n-1}} = H(e_{n-1})$ to the entry e_n. An entry is created on every period of time, i.e. 4 minutes to show a gradual process of creating the `GSBF`s. The structure of `SBFChain` is shown in Figure 5.5.

An entry e_n contains the following information:

- The hash value $h_{e_{n-1}}$.
- The block number n.
- A timestamp ts_n.
- The hash values of the most recent `GSBF`s.
  - $h_{GSBF_k} = H(GSBF_k)$.
  - $h_{GSBF_m} = H(GSBF_m)$.

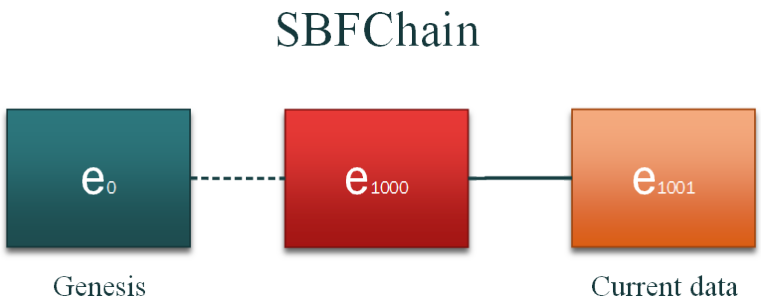

Figure 5.5: The structure of SBFChain.

- The metadata of all blockchain branches in which the information is added to the most recent SBFs.
  - Chain_ID.
  - Block Height.
  - Data Count.

By referring to the most recent entry e, one can determine which information has been added to the recent GSBFs. The entry e will also help reconstructing the GSBFs at any given time by referring to information stored in the nearest entry e.

5.5.2.3 Joint Node

We coin the term `joint node` to describe a new type of node, which stores and manages GSBFs and SBFChain. The `joint node` will be operated under a collaboration between the maintainers of multiple blockchain branches. The idea of the `joint node` originally came from `blackball` databases, where the information is collected from multiple parties [35]. At the same time, the joint node also behaves similar to hard fork oracle [71], in which information about multiple chain forks is manageable in one place. The `joint node` collects all related information from different blockchain branches and constructs SBFs and SBFChain.

The SBFChain synchronises SBFs maintained by different joint nodes. There exists a simple consensus method among the joint nodes to add new entries in the SBFChain, where every information update in the SBFChain is followed by all joint nodes as the members of the system.

The joint node will not cause any scalability issue to the main application of each blockchain, especially related to providing necessary storage and computing power to process requests and responses. Joint nodes form a new Monero subsystem. The new subsystem is different from the primary system that consists of normal nodes running Monero protocols. Although joint nodes and normal nodes are in different systems, there exists a mechanism such that the nodes can exchange information.

RPC can be used as a communication scheme between normal nodes and joint nodes as well as between joint nodes and the SPV wallets on the client-side. The P2P communication scheme is necessary for the joint nodes to update new information from the network of multiple blockchains. Figure 5.6 shows the relationship between joint nodes, normal nodes, and SPV wallets.

The users' wallets can also actively seek advice from the joint nodes regarding the raw transactions the wallet create such that the problems of key reuse prevention are on early stage. However, the normal nodes can utilise the SBFs maintained by the joint nodes to perform a simple checking algorithm before processing the transaction.

Although the joint nodes store the GSBFs, they cannot extend any blockchains nor modify the information inside the blockchains. All updates on the blockchains and memory pools will be inserted into the respective LSBFs and GSBFs. We use the term service subsystem to refer to a network of joint nodes.

5.6 Discussion

In this section, we discuss security analysis and performance analysis of our proposed solution.

5.6.1 Security Analysis

5.6.1.1 Active Attack

We assume there exist dishonest joint nodes in the service subsystem where the majority of the node members are behaving honestly by following the protocol correctly. When the dishonest joint nodes receive requests from the client (either a wallet or a normal node) to verify whether key images or key image-mixin tuples are in the current SBFs, the dishonest joint nodes will produce incorrect responses. To mitigate the problem, the client can send requests to multiple joint nodes in the subsystem selected at random. Assuming

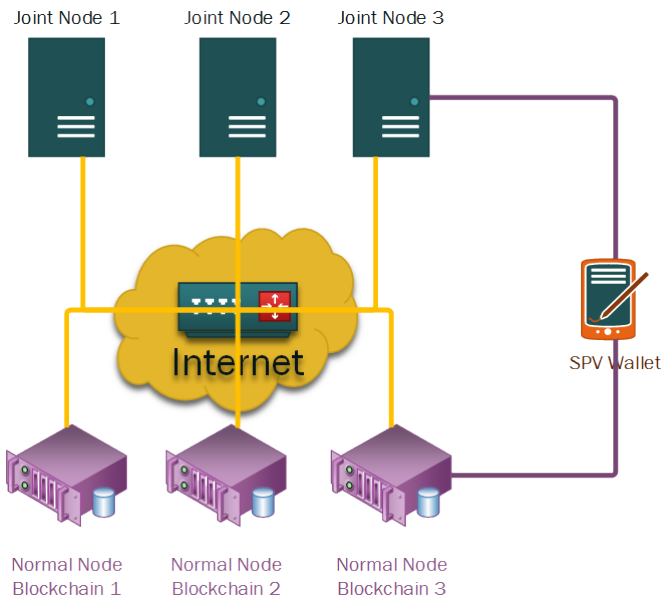

Figure 5.6: Joint nodes can assist SPV wallets as well as normal nodes of different blockchains.

that the majority of the `joint nodes` in the subsystems are behaving honestly, the client will find inconsistencies of the responses. The client then regards the results as votes to distinguish the correct responses form the incorrect ones, where the correct responses are likely to become the majority as honest `joint nodes` always return correct responses.

A dishonest `joint node` can also be detected by its peers. The `joint nodes` validate each other's `SBFs` files by confirming the hash values of the `SBFs` and the hash values stored in the `SBFChain`. If the information does not match, any nodes returning incorrect information can be blacklisted. All clients then publish the blacklist information. Random requests are useful for checking mechanism to detect any dishonest `joint nodes`.

The normal nodes of different blockchain branches can also cooperate to verify the correctness of the `SBFs` maintained by the `joint nodes`. However, this requires extra computing resources by the normal nodes. The verification of the correctness of `GSBFs` consists of two-stage reconstruction.

1. **Stage one: intrachain reconstruction**. In this stage, the normal nodes of each blockchain branch compute `Local SBFs` (`LSBFs`) by using their own blockchain data according to an agreed entry on the `SBFChain`. The reconstruction of the `Local SBFs`

can start from the `Fork Point` of that blockchain branch instead of from the genesis block (block number zero). The correctness of the `Local SBF`s (`LSBF`s) depends on the honesty of the normal nodes of the blockchain. Assuming that the majority of the nodes behave honestly, then the correct `LSBF`s can always be generated.

2. **Stage two: interchain reconstruction**. The nodes of different blockchain branches cooperate to generate a set of `Global SBF`s (`GSBF`s). These `GSBF`s are created by concatenating all `LSBF`s. Assuming that all `LSBF`s are correct, then the produced `GSBF`s is also correct.

A dishonest normal node can also try to confirm transactions that have problems of key reuse into new blocks it produces by collaborating with miners that have sufficient computing power. In this case, other normal nodes can revalidate these transactions with the help of `joint nodes`. When these transactions are proven to be malicious, then the blocks containing these malicious transactions can be ignored. Assuming that the majority of the nodes behave honestly, then there will be a temporary fork which will resolve after several blocks according to the current Monero protocol. Since the miners will suffer financial loss if the system removes their produced blocks, they are less motivated to behave dishonestly.

5.6.1.2 Passive Attack

In the passive attack, the attacker has access to the public blockchains. The attacker develops analytic tools to determine traceable transactions. The attack depends on the number of traceable transactions and the portion of the traceable transactions compared to all transactions.

Our proposed solution can prevent active and passive attacks. Passive attacks analyse existing valid transactions in the blocks. With no addition of malicious transactions to the blocks, then the passive attack will not produce any expected outcome, assuming that the attacker does not receive any extra information.

5.6.2 Performance Analysis

5.6.2.1 Hard Fork Management

We conducted experiments to calculate the extra computing resource in managing the extra information for hard fork management. The experiments used a Ubuntu 18.04 LTS virtual

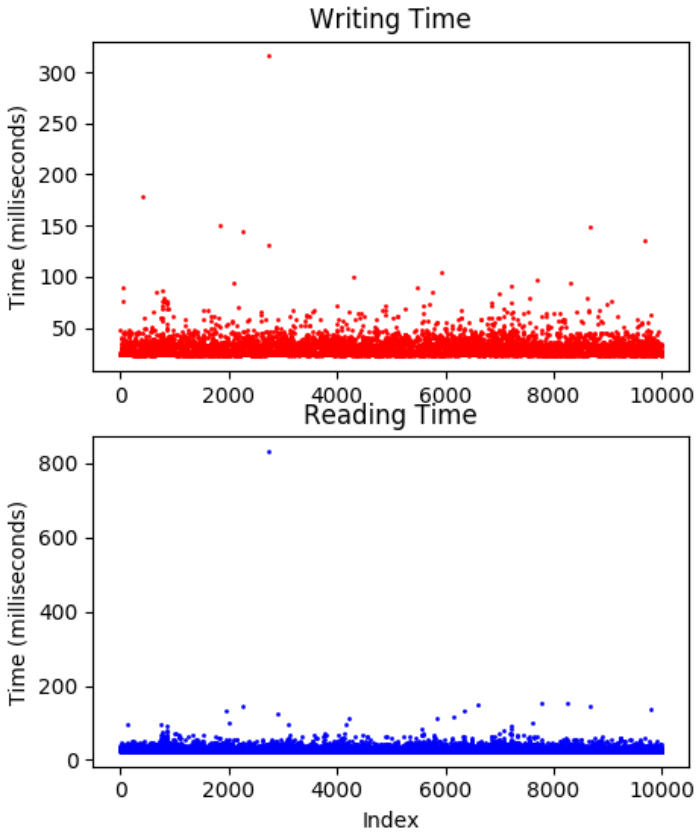

Figure 5.7: The read-write processing time for `Chain_Info` database using `LMDB`.

machine equipped with 8GB RAM and maximum 2 CPU cores. We created a new table called `Chain_Info` using `LMDB` database system, which is the same database product that stores and manages the blockchain data of the current version of Monero.

Two million `Chain_ID` - `Fork_Point` tuples were written to the database and then read. We repeated the process 10,000 times. About 1.2MB storage was required to store the two million records while writing average time was 28.37 milliseconds, and the average reading time was 28.19 milliseconds. Figure 5.7 shows the detailed result of the experiment. The experiment shows that the computing resource for the required operations is insignificant such that today's regular computers can afford it.

5.6.2.2 Joint Node Affordability

In our proposal, a particular type of node called `joint node` maintains the `GSBF`s. The set of `GSBF`s which are relevant to all existing or future Monero blockchain branches.

We experimented with calculating the time and storage needed to create a `SBF`. The experiment utilises Jay Baird's Scalable Bloom Filter Python library, `pybloom`[8]. The experiment used a `LARGE_SET_GROWTH` setting to anticipate considerable dataset growth. In this setting, a surge jump in storage size will happen every time the system hits its maximum capacity. The results as in Figure 5.8 show that the time required to create a `SBF` is linear to the number of the data inserted in the `SBF` with the average of 17.308 data per second. The file size, however, increased significantly every time the capacity is full, according to the `LARGE_SET_GROWTH` algorithm. Our experiment also showed that creating a `SBF` with 100 million data produced 372.1MB of `SBF` file within around 96 minutes. Due to the low resource requirement when creating the `SBF`, recalculating the `SBF` will not be a problem.

5.6.3 Limitation

By mitigating the problem of key reuse, our solution can mitigate a passive attack which utilises analyses on public blockchains. Our solution cannot prevent a passive attack on the network level, which is one of the biggest privacy issues in cryptocurrency [46]. Our solution is also prone to a passive attack conducted by an honest-but-curious `joint node`, where the `joint node` can potentially trace users' transaction given enough information. This problem, however, is not exclusive to our proposed system, but also applies to all Monero nodes.

5.7 Conclusion and Future Work

We investigate the problem of key reuse as an unwanted impact of Monero hard forks. We build a dataset from three different blockchain branches and determine the traceable inputs as the result of the key reuse problem. We also identify the cascade effect and the reduced anonymity as the side effects of the main problem. Our analyses discover that the scalability of the problem of key reuse correlates to the market price of the respected coins. The support from cryptocurrency markets to new coins is also an important factor in the

[8] https://github.com/jaybaird/python-bloomfilter

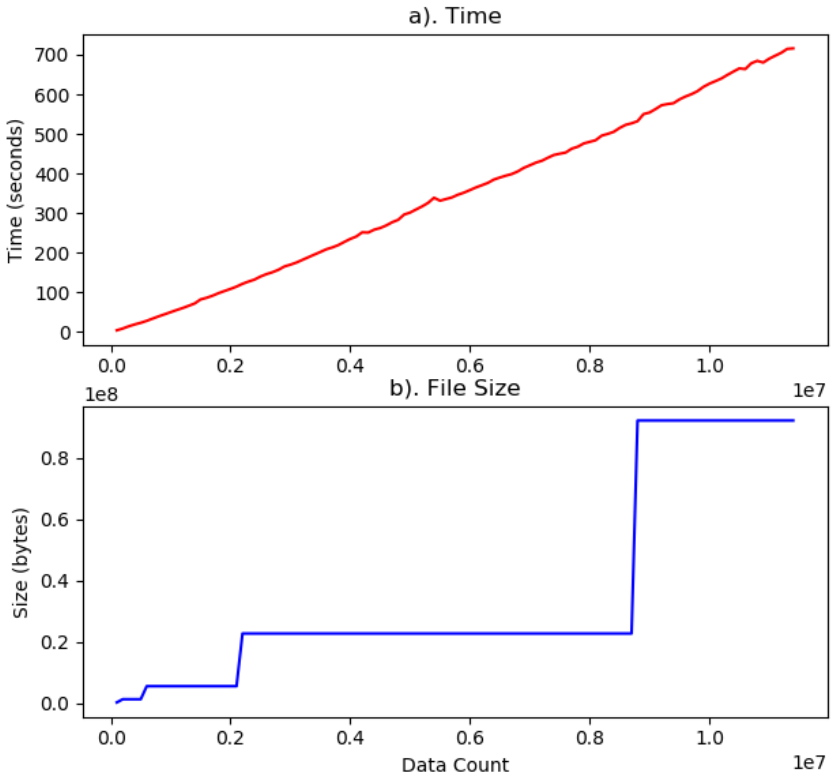

Figure 5.8: Part a) shows the creation time of `SBF` which is a positive linear to the data size. Part b) shows the result's file size where the file size will be increased when the capacity of the `SBF` is full.

problem of key reuse. We propose a mitigation strategy for hard fork management and key image management, where joint nodes play an important role in the proposed strategy.

For future work, we will investigate how our solution works in different types of cryptocurrency. We will also investigate different options in handling protocol-level changes to avoid a hard fork. The new method should be able to support fundamental changes in the system without creating a new blockchain branch. This type of solution is useful in systems with active development such as Monero. It is also interesting to further investigate the correlation between the cryptocurrency market price and the number of transactions recorded in Monero blockchain to uncover the actual behaviour of Monero users. The use of Monero in the real world is also a topic of interest.

5.8 Chapter Summary

In this chapter, we investigate the impact of Monero protocol updates that caused blockchain hard forks in 2018. We discover that Monero becomes vulnerable to traceability attacks during a hard fork. Our findings expose traceable transactions on each Monero blockchain branch on two subsequent hard forks in 2018. We propose mitigation strategies that require collaboration from system maintainers of all blockchain branches after a hard fork.

We investigate another problem during Monero protocol update in the next chapter. We analyse how an attacker conducts a Denial of Service (DoS) attack to non-updating nodes. We also show an escalated DoS attack into transaction anonymity attack.

Chapter 6

Denial of Service and Traceability Attacks on Monero Protocol Updates

A cryptocurrency protocol incorporated in a node application runs without human intervention. The protocol utilises cryptographic techniques to determine the ownership of the coins. The cryptographic techniques also enable coin ownership transfers between users. Consensus protocols determine the source of the truth of the information contained in the public ledger called a blockchain.

When the protocol needs to be updated, all nodes need to replace the application with the newest release as soon as possible. However, the nodes' update time varies since they belong to different parties. Hence, Monero protocol update is an asynchronous event. A Monero protocol update also causes a blockchain hard fork, where a new branch emerges. The new branch is incompatible to the old branch because they run different sets of protocols.

In this chapter, we continue to work on our second research question, `Q2`, that aims to identify alternative threats to Monero system that exploit the Monero asynchronous protocol update. We explore the asynchronous protocol update that creates a vulnerability in Monero nodes which have not yet updated to the newest software version.

We show that an attacker can launch a Denial of Service attack against the nodes running the outdated protocol, where the attack significantly reduces the system' performance. We also show that an attacker with sufficient access to cryptocurrency services can escalate the Denial of Service attack to anonymity attack. We develop this chapter based on our published paper, *Risk of Asynchronous Protocol Update: Attacks to Monero Protocols* [118].

6.1 Introduction

There are usually two main applications in a cryptocurrency systems, including Monero. These two applications are node application (node) and wallet application (wallet). Monero node stores and maintains the Monero blockchain. A Monero node is connected to other nodes through a peer-to-peer network. Monero node also stores unconfirmed transactions in its temporary database called transaction pool (`txpool`) located in the node's RAM. Monero's (`txpool`) is identical to `memory pool` (`mempool`) in other cryptocurrencies such as Bitcoin [5]. The unconfirmed transactions are stored in the `txpool` for at most three days, or `(86400*3) seconds` as defined in `DEFAULT_TXPOOL_MAX_WEIGHT` parameter in `src/cryptonote_config.h` [54].

Monero wallet (or wallet) is the application on the client-side. A wallet helps a user manage her private keys, track remaining balance, detect incoming transactions, and create outgoing transactions to spend coins. Monero wallet is a thin client; it does not store any information about the blockchain. Therefore, for each operation on the Monero wallet that requires blockchain information, the wallet will connect to a Monero node.

Monero Classic emerged as the result of Monero hard fork in April 2018. While the Monero main chain upgraded to Monero protocol version seven, Monero Classic runs on Monero protocol version six. On 16 October 2018, Monero Classic announced a protocol upgrade [25]. The upgrade intended to add more features in Monero Classic system [25]. Furthermore, at the same time, Monero Classic increased the minimum ring size from five to eight. The change in minimum ring size impacted the protocol as it caused a **protocol reduction** [126], but due to its circumstances, no hard fork occurred after the event. Protocol reduction is a type of protocol change where the new protocol reduces the rules of the old protocol. Protocol reduction is usually followed up by a hard fork if miners support both protocols with reasonable computing power.

In this chapter, we propose attacks to the transaction pool of Monero nodes running an old protocol after a protocol reduction event. An attacker inflates the transaction pool size through a Denial of Service (DoS) attack to the nodes. The attack significantly reduces the quality of service of the nodes under attack. We also expose that the attacker can escalate the DoS attack to reveal the users' traceability by double-spending the same coins in two different protocols.

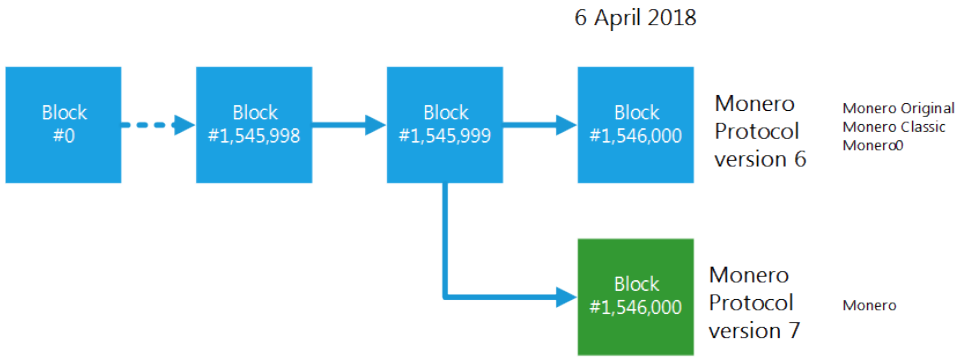

Figure 6.1: Monero hard fork in April 2018.

6.2 Background

6.2.1 Monero Hard Fork

Unlike other cryptocurrencies, which always try to avoid hard fork, Monero has scheduled semi-annual hard forks to improve the system. Before 2018, there were already five Monero hard forks for protocol upgrades [75]. The sixth protocol upgrade which occurred on 6 April 2018 split the Monero blockchain into two branches; the first branch ran Monero version six (the old protocol) and the second branch ran Monero version seven (the new protocol). While the latter branch became the main Monero branch, the other branch running Monero protocol version six has different names: Monero Original[1], Monero Classic[2], and Monero0[3] [74]. All of the mentioned names are less popular than the Monero main branch (XMR). Monero Classic (XMC) and Monero Original (XMO) were tradeable in several cryptocurrency exchanges[4]. Figure 6.1 shows the Monero hard fork that occurred in April 2018.

[1]https://monero-original.org
[2]http://monero-classic.org
[3]https://monero0.org
[4]XMO was no longer available in the market as of 12 February 2019. However, the market price history provided by Coinmarketcap.com shows that XMO were traded until 1 February 2019. Based on Coinmarketcap.com, XMC is currently available in Gate.io, HitBTC, and TradeOgre.

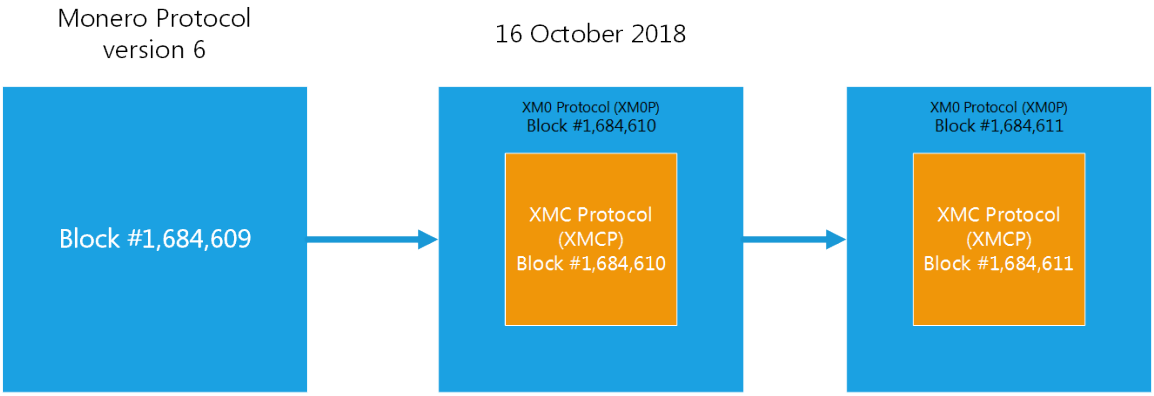

Figure 6.2: Monero protocol version 6 hard fork in October 2018.

6.2.2 Monero Classic Protocol Upgrade

We define a Monero0 Protocol `XM0P` as the protocol which accepts transactions with a minimum ring size $r \geq 5$. The protocol `XM0P` is the same protocol as Monero Protocol version 6 before Monero Classic upgrade. We also define a Monero Classic Protocol `XMCP` as the new protocol which only accepts transactions with a minimum ring size $r \geq 8$. The protocol `XM0P` allows a new block b' from the protocol `XMCP` to be included in the blockchain C, but the new protocol `XMCP` does not allow a new block b from the protocol `XM0P` to be in the blockchain C' such that $C \supset C'$ [126]. This occurs if the block b contains any transactions where $5 \leq r < 8$. Figure 6.2 shows how blocks created by using `XMCP` protocol is still compatible with `XM0P` and it shows that `XMCP` reduces the rules of `XM0P`.

Monero Classic claimed to acquire the majority of the miners. Therefore the new protocol `XMCP` applies to the blockchain branch running Monero protocol version six. Since the protocol upgrade, there was no record of the transaction from protocol `XM0P` in the blockchain. Although the protocol upgrade was announced prior to October 2018, the Monero0 nodes running the protocol `XM0P` were still operational until the end of January 2019[5]. As the result of the minimum ring size increase, the transactions which had a ring size of less than eight were incompatible with new blocks. The transactions stalled in Monero0's nodes' transaction pool.

From November 2018 until the end of January 2019, we discovered 115 transactions that contain 337 inputs and 1,865 ring members that used a minimum ring size of $5 \leq r < 8$

[5]According to Monero0.org, the Monero0 nodes are `159.65.227.38`, `167.99.96.174`, `159.65.113.142`. Based on our investigation, all of these nodes were no longer accessible as of early February 2019.

in Monero0's temporary storage. As the result of the incompatibility between Monero0's protocol `XM0P` and Monero Classic's protocol `XMCP`, these transactions were not on Monero Classic node's temporary storage. No miner ran the protocol `XM0P`. The system never confirms the 115 transactions, and the blockchain hard fork never occurred. We also discovered different transactions used identical inputs which contain different sets of mixins or decoys and different ring size. From this occurrence, we discovered five traceable inputs.

6.2.3 Denial of Service Attack in Cryptocurrency

In cryptocurrency space, Denial of Service (DoS) attack is one of the challenges for cryptocurrencies [15]. DoS attacks target several services, such as cryptocurrency exchanges, mining pools, gambling services, financial services, wallet services, and others [111]. Not only attacking services, but DoS attacks can also be launched on the cryptocurrency to disrupt the system, e.g. by creating a large number of transactions, each contains dust coins [17, 5].

A DoS attack scenario to target Monero's `txpool` has been discussed in [27, 26]. In the proposed scenario, the author described that transactions with the size of 360kB could be created by modifying `monero-wallet-rpc` source codes [27, 26]. These transactions have a low priority but contain enough transaction fee to relay. However, due to the median block size growth protocol, the author stated that the system could not include the transactions to the blockchain. Hence, it would be on Monero node's `txpool` for a maximum allowed time [27, 26].

The author reported the problem to Monero developers as a potential vulnerability. The developers then acknowledged the problem and provided an optional setting to limit the maximum `txpool` size[6]. However, when we further investigated the matter, we could not find the related codes in the newest Monero software. In this vulnerability report, the author did not provide detailed information on the attack, what modification made, nor whether the attack was successful.

[6]https://github.com/monero-project/monero/pull/3205

6.3 Related Work

6.3.1 Cryptocurrency Protocol Change Classification

Zamyatin et al. [126] classified four types of protocol changes from the old protocol P associated with a blockchain C to the new protocol P' associated with a blockchain C'. These changes are: **protocol expansion**, **protocol reduction**, **protocol confliction**, and **conditional protocol reduction** [126]. The protocol expansion occurs if the new protocol P' increases the previous protocol P's scope such that the set of blocks V' of the new protocol P' expands the set of blocks V of the previous protocol P. On the other hand, a protocol reduction occurs if the new protocol P' reduces the previous protocol P's scope by adding more rules. Protocol confliction occurs if the new protocol P' is incompatible with the previous protocol P, while the conditional protocol reduction (velvet) is a protocol reduction on specific elements of the protocol, where no changes is supposedly to happen in the created blocks of both protocols P and P'.

6.3.2 Key Reuse Attack in Monero

The Monero fork which occurred on 6 April 2018 has created a new attack vector called key reuse [31]. A new traceability analysis called cross-chain analysis was developed and found that 5.22% of the transaction inputs created between 1 April 2018 and 31 August 2018 are traceable [47]. Although the result of the attack is insignificant compared to other attacks such as zero mixin attack, similar events in the future can occur under a certain condition. The Monero developers have responded the issue by implementing `Shared Ringdb` feature in the default Monero wallets, including `monero-wallet-cli` and `monero-wallet-rpc`.

6.4 Threat Model

It is assumed that there exist two nodes, $Node_A$ and $Node_B$, running two different protocols, P and P'. The protocol P' is an upgrade from P, where the upgrade is a protocol reduction as described in [126]. It is also assumed that all miners have already updated their protocol from P and P' and connect to $Node_B$.

We propose the threat model as follows. An attacker owns sufficient funds and can create valid transactions under P but invalid under P'. The attacker launches Denial of Service attacks by creating a set of transactions T as many as possible such that the transactions

T remain in the $Node_A$'s `txpool`. The attacker has access to two wallets: $Wallet_A$ and $Wallet_B$. $Wallet_A$ is connected to $Node_A$ and $Wallet_B$ is connected to $Node_B$. The attack is successful if the resource usage of $Node_A$ increases significantly compared to the $Node_B$ as the control.

We assume that the attacker has control over a cryptocurrency-related service such as a coin exchange. In this scenario, the attacker creates a set of transactions T' from the same coins that have been spent in T to be double-spent in T'. The purpose of this activity is to let any observers conduct traceability analysis as described in [47] by comparing two sets of transactions T and T'.

6.5 Attacks to Monero Protocols

6.5.1 Overview

The attacks were developed based on the problem discovered in section 6.2.2. The attacker increases the scale of the attack to create more severe damage to the node and the transaction anonymity.

The attacks exploit the late update conducted by node maintainers when there is a change in the protocol where every node should update its software to the latest version. If the change is a protocol reduction, the old version of the protocol accepts new transactions. However, the new version of the protocol will not be able to confirm transactions that follow the old protocol. As a result, an old node keeps these incompatible transactions in the `txpool` for at least three days.

An attacker floods the `txpool` with incompatible transactions which will create a denial of service (DoS) to the nodes running the old protocol. The massive number of transaction creation and the large transaction storage requirement will reduce the performance of the nodes. Since the blockchain will never confirm the transactions, there will be no transaction fee the attacker needs to pay. However, the attacker needs to have sufficient funds in as many outputs as possible.

6.5.2 Attack Phases

The attack consists of three phases: the preparation phase, Denial of Service (DoS) Attack phase, and traceability reveal attack phase. The attacker can conduct the last two phases separately or subsequently.

6.5.2.1 Preparation Phase.

In the preparation phase, the attacker first prepares enough funds. Then, the funds will be sent to as many public keys (outputs) as possible by creating transactions to send the funds to the attacker's address. The transactions need to be compatible with the new protocol; the nodes that run the old protocol will also confirm the transactions.

6.5.2.2 Denial of Service Attack (DoS) Phase.

In the attack phase, the attacker launches the attack against the old nodes. The attacker creates as many transactions as possible, where the new protocol cannot confirm these transactions because it follows the old protocol, which is incompatible with the new protocol. The transactions are likely to be discarded by the new nodes. However, the old nodes will store the transactions in the `txpool`. When the number of transactions is large, the `txpool` size will expand significantly and affect the system.

6.5.2.3 Traceability Reveal Attack Phase.

If the attacker has control over a cryptocurrency service provider, the attacker can create transactions to spend the same coins used in the DoS phase. Assuming that the transactions follow the service provider' best interest, the attacker would not need to worry about losing money for the transaction fees, because the customers pay the transaction fees.

6.5.3 Simulation

We simulated the protocol reduction event as described in Section 6.2.2 in a private `testnet` environment. The simulation scales the event to analyse the impact of a large scale attack. Instead of using Monero Classic and Monero0 nodes, we modified Monero *Beryllium Bullet* software[7]. The latest upgrade of Monero requires a mandatory bulletproofs feature for every transaction to be validated in the node to reduce the transaction size significantly. We also ran Monero blockchain explorers called `Onion Monero blockchain explorer` (`xmrblocks`)[8] to simplify the data extraction process from the nodes. The blockchain explorers were also used to monitor the size of the `txpools` of the nodes. The `xmrblocks`

[7]The open-source software is available in Monero's Github page https://github.com/monero-project/monero.

[8]https://github.com/moneroexamples/onion-monero-blockchain-explorer

would only be run during the data extraction process at the end of the simulation to reduce the extra workload that might affect the result.

We ran our simulations on a `Ubuntu 18.04.1 LTS` virtual machine with 8GB RAM and two processor cores. We hosted the virtual machine was hosted on a physical server equipped with two Intel® Xeon(R) CPU X5570@2.93GHz and 72GB RAM. The virtual machine ran two `monerod` as the Monero nodes and two `xmrblocks` applications, each connected to a different node. We used a default RPC-based wallet, `monero-wallet-rpc` connected to the target node. We wrote a Python script to automate the transaction creation with the help of Shoupn's `moneropy` library[9] to connect the script to `monero-wallet-rpc`.

6.5.3.1 Simulation Scenario

Two nodes, denoted as $Node_A$ and $Node_B$, were used to construct the Monero network. Both nodes synchronised their data with a blockchain C. $Node_A$'s source code was modified such that the node could validate any transactions with a ring size $r \geq 7$, while $Node_B$'s protocol mimicked the latest Monero transaction requirement with the ring size $r = 11$. These two nodes imitated a situation of a **protocol reduction** from protocol P_A to protocol P_B, where the new protocol requires a larger ring size such that $r_A < r_B$

If all miners have updated their software to follow the new protocol P_B, protocol P_A does not have enough mining power to compete with the new protocol P_B in creating blocks. However, the node $Node_A$ still follows protocol P_A while the node $Node_B$ has updated to protocol P_B.

6.5.3.2 Simulation 1

The first simulation evaluated a key reuse attack which can potentially happen in the given scenario. Although a key reuse attack normally happens in two or more different blockchains, the given scenario can also create two temporary blockchains, where the `txpools` of the nodes pose as the temporary blockchains.

We created 9,353 transactions T_1 with the ring size $r = 8$. Each contains 10 or 11 outputs (transactions which have 11 outputs contain one address to send the change money back to the sender). There were a total of 18,688 inputs in transactions T_1, and the total size is 34.2MB. The transaction fee required to create all transactions T_1 is 80.46XMR, where the average transaction fee is 0.0086 per transaction. As a result, the transactions

[9]https://github.com/shoupn/moneropy

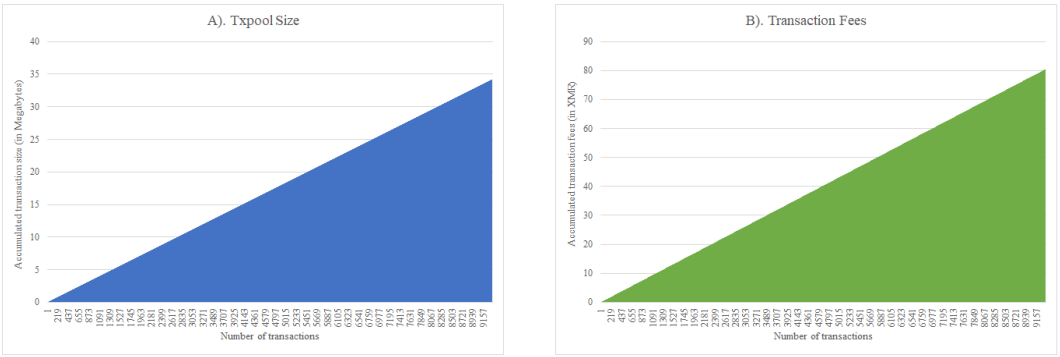

Figure 6.3: Figure (A) shows the accumulated transaction size stored in `txpool`. Figure (B) shows the transaction fee paid by the attacker to create the transactions.

would be validated only by the $Node_A$ but not $Node_B$. The transactions T_1 were stored in $Node_A$'s `txpool`. However, none of the transactions T_1 were found in $Node_B$'s `txpool`. Figure 6.3 shows the linear relationship between `txpool` size and the number of transactions. Also, we discovered a similar linear relationship between the transaction fee and the number of transactions.

A new set of transactions T_2 were created by using the protocol P_B from the same pool of coins. In this case, there is a possibility of sending the same coins in two different transactions t_1 and t_2 where $t_1 \in T_1$ and $t_2 \in T_2$ and $T_1 \neq T_2$, since none of the transactions in T_1 were confirmed in the blockchain. Transactions T_2 were accepted and confirmed by $Node_B$ as the miner and then $Node_A$ had the same information after synchronising with $Node_B$.

6.5.3.3 Simulation 2

In the second simulation, we flooded the $Node_A$ with transactions. We utilised the standard Monero RPC-based wallet, `monero-wallet-rpc` which was connected to $Node_A$ to programmatically created Monero transactions using a single thread. We ran the script simultaneously. We then evaluated the performance of $Node_A$. When the data was evaluated, there were 64,633 transactions in $Node_A$'s `txpool`.

6.6 Discussion

6.6.1 Shared Ringdb

During our first trial of **Simulation 1**, we were unable to recreate the key reuse attack. All transactions T_2 which spent the same coins as T_1 used identical ring members as t_1. The transactions in T_2 added other outputs to satisfy the minimum ring size of the protocol P_2. We evaluated the occurrence and discovered that the newest version of Monero default desktop wallet took a preventive action by creating an additional database called `Shared Ringdb` located in `~/.shared-ringdb` folder. The wallet can detect the difference between two systems based on the protocols, where a transaction that tries to spend the coin that has appeared on another blockchain needs to maintain the anonymity set by using the identical ring members.

This result shows the effectiveness of `Shared Ringdb` for general users who run the Monero default desktop wallet and deserve better anonymity in their cross-chain transactions. However, the feature is infeasible to implement on alternative wallets such as web wallet or smartphone wallet. An attacker can also simply remove the database in the attacker's local storage.

6.6.2 Traceability Analysis

An input in Monero is traceable if an observer manages to guess the real spent output over a set of outputs in that input with the probability of `one`. Based on the specification given by the Linkable Ring Signature [64], each private key has a unique tag called `key image`. If a `key image` appears in two or more ring signatures, then it means that the same private key signs these signatures, and the associated public key must be a ring member of both ring signatures. Therefore, the public key has become traceable. This type of analysis has been discussed by Hinteregger et al. [47].

After the **Simulation 1** concluded, a traceability analysis followed by using an algorithm as follows.

1. For each input in $Node_A$'s `txpool`:
   (a) Identify the key image value.
   (b) Find the `key image` in the blockchain.

(c) If the `key image` is successful, proceed to the next step. Otherwise, continue to the next input.

(d) Examine the related sets of outputs. If at most one identical output is found on multiple sets of outputs, then the input is traceable.

(e) Otherwise, the input is not traceable. However, the effective ring size (the term coined in [47]) is reducible to exactly the number of outputs found on both sets.

2. The result is the traceable inputs identified by the algorithm.

By using the above algorithm, we then evaluated the 9,353 transactions we created and managed to identify `95%` of the transactions' inputs. The other `5%` of the inputs suffer reduced effective ring size to a minimum of two. The result shows the effectiveness of the method in revealing Monero transaction's traceability.

6.6.3 Denial of Service Analysis

We define a `Quality of Service` (`QoS`) of the cryptocurrency system (the node and the wallet) as the number of new transactions created by the wallet and received by the node. The `QoS` was measured to determine the impact of `DoS` attacks. We also measured the resource usage (CPU and RAM) of both $Node_A$ and $Node_B$.

6.6.3.1 $Node_A$'s Quality of Service Monitoring.

Figure 6.4 shows the `QoS` of $Node_A$, measured by number of new transactions per hour. It shows that after the `txpool` of $Node_A$ exceeds 7,800 transactions, the `QoS` sharply declines from the peak, 3,000 transactions per hour (tph), to below 500 tph when there are at least 53,000 transactions in the node's `txpool` and the total size of these transactions is `190MB`. These 53,000 transactions requires `375.6XMR` as the transaction fees. However, as these transactions will not be confirmed in the blockchain, the cost does not really incur. The highest `txpool` size was `457MB`.

6.6.3.2 Resource Usage Monitoring.

We also ran a simple script to capture information from `Ubuntu`'s `top` command every 60 seconds. Figure 6.6 shows the comparison of CPU usage (measured in percentage of usage compared to the system's CPU) between $Node_A$ and $Node_B$, where $Node_A$ was a target to

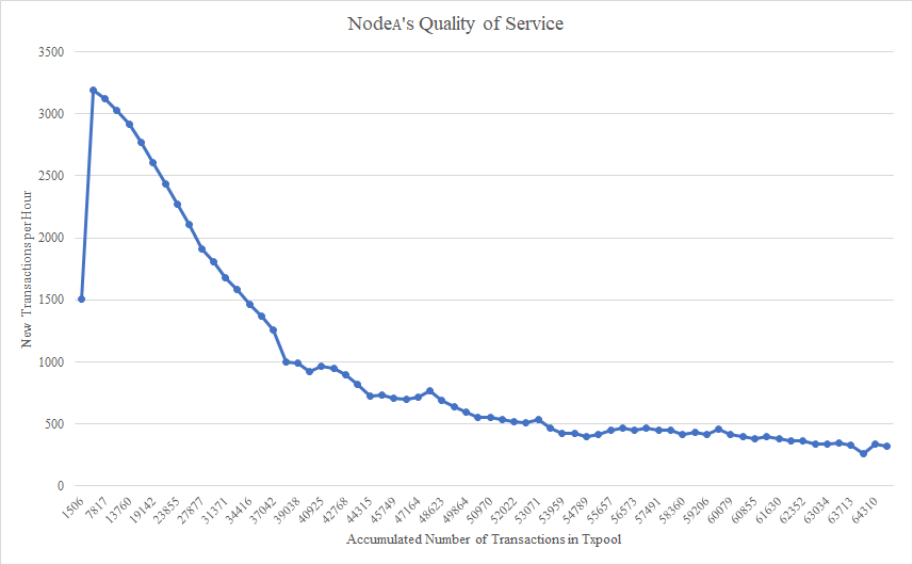

Figure 6.4: The `Quality of Service` of $Node_A$ during the DoS attack.

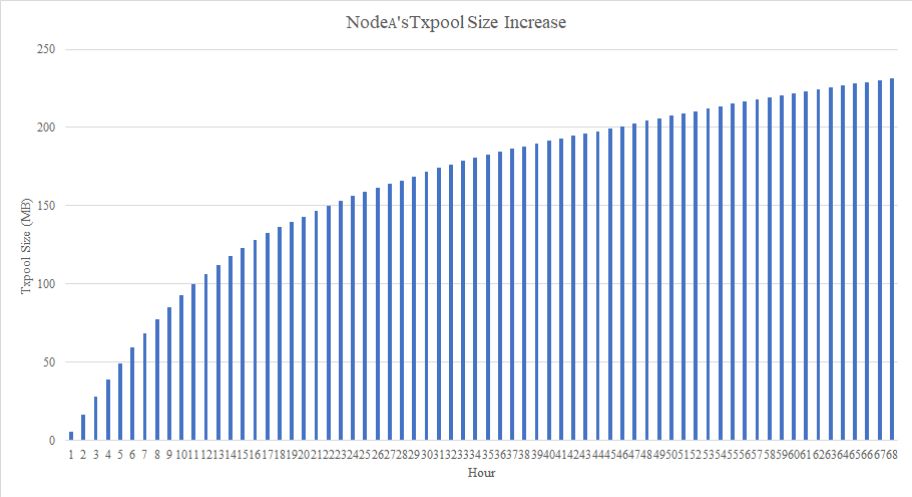

Figure 6.5: The `txpool` size increase of $Node_A$ during the DoS attack.

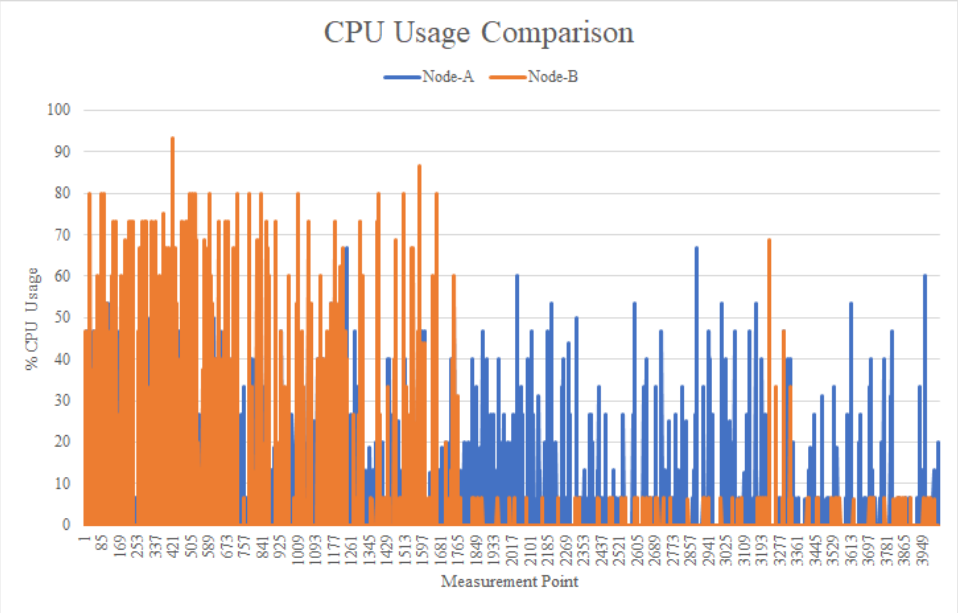

Figure 6.6: The CPU usage comparison between $Node_A$ and $Node_B$.

the DoS attack and $Node_B$ was not. In this scenario, none of these nodes was mining new blocks. The result demonstrates an insignificant difference between both nodes in terms of CPU usage.

Figure 6.7 shows the comparison of RAM usage between two nodes, where $Node_A$'s RAM utilisation was four times more than $Node_B$'s RAM utilisation. The highest RAM size used by $Node_A$ was near to `9%` of the system's RAM, or roughly about `720MB`.

6.7 Limitation

In our experiment, we used one virtual machine to run multiple applications, such as node applications, wallets, blockchain explorers, and the monitoring script. The low computing resources we used could have impacted the performance of the evaluated node. The results might differ if these applications run on separate machines. Our experiments did not utilise the Monero version 6 source code (Monero Classic or Monero0), as we preferred to use the latest Monero applications. The version selection may or may not impact the result, since the transaction size would be different compared to our result, including the fee structure.

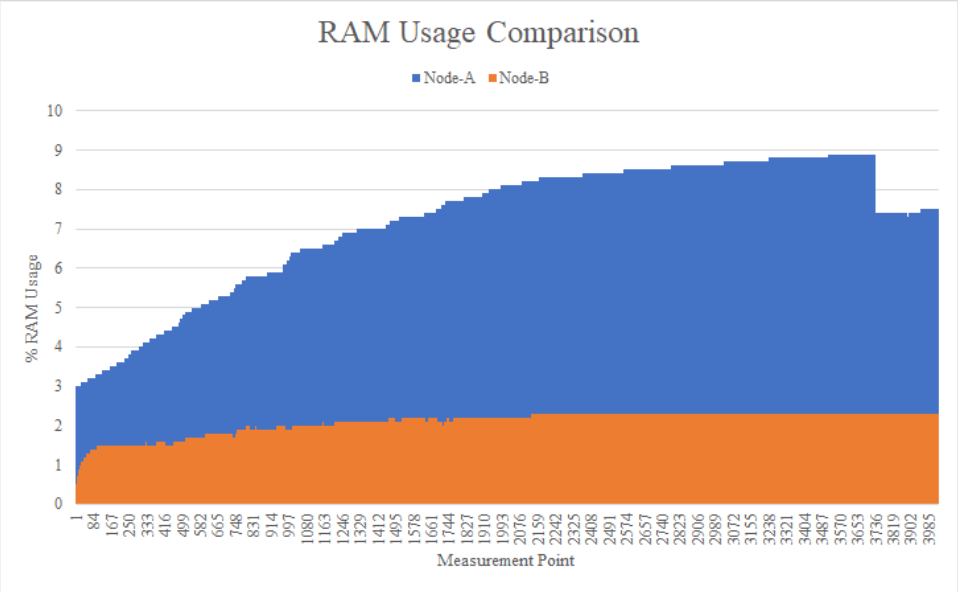

Figure 6.7: The RAM usage comparison between $Node_A$ and $Node_B$.

6.8 Conclusion and Future Work

We present attacks to `txpool` of nodes running an outdated protocol P when almost all participants in the system have updated to a new protocol P', where the protocol P' reduces the scope of protocol P. Through simulations, we show that this event can be exploited by an attacker to launch a Denial of Service (DoS) attack to the nodes running protocol P. The DoS attack reduces the `Quality of Service` (`QoS`) of the target node, where the number of new transactions served by the target node is significantly reducible when the `txpool` increases.

The attacker can escalate the event to trace users' transactions if the attacker controls a cryptocurrency service such as coin exchanger. It is done by double-spending the coins, where the attacker creates two sets of transactions: each set for different nodes running the protocol P and P'.

For future work, we suggest the investigation of other types of protocol changes and how they impact nodes that do not update their systems to the latest versions. The result of the investigation can help to formulate a better protocol update mechanism to protect the nodes running on old protocols.

6.9 Chapter Summary

In this chapter, we continue to work on our second research question, Q2, that focuses on identifying threats to Monero system that exploit the Monero asynchronous protocol update. The protocol update requires each Monero server to update its running Monero software as soon as possible. However, we discover that if a server delays the software update, it becomes vulnerable to Denial of Service (DoS) attack. We also show that the attacker can escalate a DoS attack to anonymity attack.

This chapter concludes our second part of the project. In the next two chapters, we work on our last part of the project. We investigate two third-party services in Monero, specifically wallet service providers and mining pools, that can become threats to Monero anonymity.

Chapter 7

Anonymity Risk of Monero Wallet Service Provider

We discuss our last research question, Q3, in two subsequent chapters. Our third research question investigates the risks for the privacy-preserving cryptocurrency that rely on third-party services to provide essential services for the ecosystem. We identify two critical third-party services in Monero ecosystem, namely wallet service providers and mining pools.

In this chapter, we examine the anonymity risks of using Monero wallet service providers when creating transactions. Cryptocurrency users are provided with wallet applications to send or to receive coins. Wallet applications connect to cryptocurrency network through nodes or wallet services. These nodes or wallet services provide blockchain data to the wallets through the Internet.

We discuss how Monero wallets are prone to anonymity attacks by third-party wallet service providers. We propose a new anonymity attack that leverage change coin spending by Monero wallets. We analysed nine wallets that are available in the market and found that the majority of the wallets are vulnerable to our identified attack. Our analysis shows that a wallet service provider can guess change coins that are spent by the wallets with high accuracy. We also propose strategies to improve the anonymity of Monero wallets by using multiple services and connection scheduling. Our manuscript, entitled Change in Transactions: A Side Channel Attack to Monero Anonymity, is the fundamental work for this chapter.

7.1 Introduction

Monero is one of the most successful cryptocurrencies in the world. At the time of writing, Monero's total market cap is nearly US$1.3 billion[1]. Monero wallet (we refer to this as a wallet) and Monero node (we refer to this as a node) are two main applications in Monero ecosystem. The nodes act as servers which provide information to wallets. The information provided by the nodes is related to Monero blockchain that is maintained by the nodes. The nodes update the blockchain data by adding new blocks or replacing previous blocks if the block reorganisation event occurs. The nodes receive these updates from other nodes by using a peer-to-peer network scheme.

The wallet, on the other hand, is a client-side application. The wallet does not maintain blockchain data, and therefore, it is sometimes called a thin client or thin wallet [62]. Although the wallet does not have the blockchain data, it needs the blockchain to operate, i.e. to update the coin balance and to create new transactions. Hence, it needs to connect to the nodes to fetch all necessary data.

Monero community recommends users to run the node locally to provide the highest guarantee of transaction privacy because a node can relate the transactions and the IP addresses of the wallets that create those transactions [60]. However, running a local Monero node requires immense Internet bandwidth, storage, and CPU. Devices with limited resources such as smartphones usually do not run any Monero nodes; therefore, wallets on those devices rely on wallet service providers that provide the required blockchain information. Our survey on Monero wallets, as shown in Table 7.1 shows that the majority of Monero smartphone-based wallets can only allow users to connect to wallet service providers to create new transactions.

Consider the following scenario. When the coin denomination is higher than the amount to send, the payer gets the change on the difference. Let Alice owns `1 XMR` in a one-time public key (we refer to this as a key) and Alice wants to pay Bob `0.1 XMR`. The only way for Alice to pay Bob is to spend the whole `1 XMR` in a transaction. The transaction splits the coin into two keys. Bob receives a key that contains `0.1 XMR`, while Alice receives `0.9 XMR` as a change coin in a new key. The change coin can be spent in Alice's next transaction, for example, to pay Carol `0.2 XMR`. The scenario is illustrated in Figure 7.1.

Alice's wallet service provider knows that one of its clients created transaction TX_1. The wallet service provider also knows that transaction TX_1 produces two coins, namely Coin B and Coin A'. Later, Coin A' is input to transaction TX_2 that produces Coin C and

[1]Based on Coinmarketcap.com as of 16 August 2019.

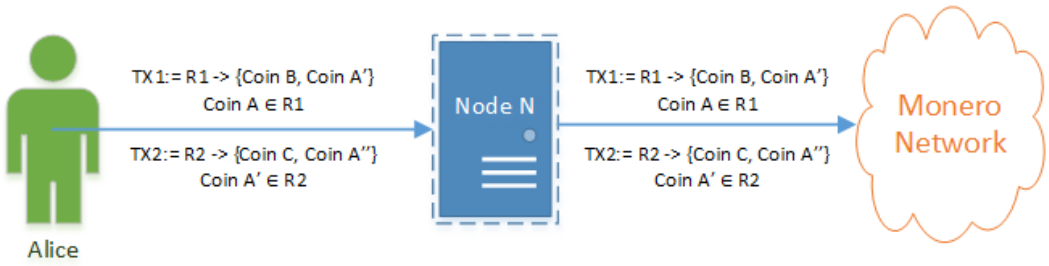

Figure 7.1: Monero transaction diagram from a user's thin wallet to Monero network through node N.

Coin A". In this work, we prove that the wallet service provider can accurately guess that Coin A' is spent either by transaction TX_2 or another transaction that includes Coin A' as one if its mixins.

Contributions. We summary our contributions as follows.

1. We propose a new passive attack where a wallet service provider as the `Attacker` guesses the spent change coins of transactions created by wallets that always connect to the same service provider. We simulated the scenario and analysed the synthetic data generated by our simulations. Our findings show that the wallet service provider can accurately guess the spent change coins in those transactions with low false-positive rates.

2. We propose several mitigation strategies to prevent the node from guessing the spent change coins, namely `Naïve Multinode Wallet (NMW)` and `Node Scheduling Wallet (NSW)`. `NMW` can reduce more than 98% of the guessed spent change coins by using more than 50 different service providers, where `NSW` completely removes the `Attacker`'s capability of correctly guess any spent keys. The number of required nodes n for `NSW` is $i + 1$ where i is the number of inputs in the transaction.

Organisation. The rest of the chapter is as follows. Section 7.2 describes background knowledge about Monero. Section 7.3 mentions current work related to Monero anonymity analyses and Monero network analyses. In Section 7.4 we present the security model that we use in this chapter, while in Section 7.5 we present how a Monero node can guess the spent keys of the transactions created by the wallets connected to it. Mitigation strategies are presented in Section 7.6. Section 7.7 briefly describes the limitation of the work, and lastly, Section 7.8 provides the conclusion and future work.

7.2 Background

7.2.1 Monero Anonymity Features

The success of Monero lies in solving privacy issues that are experienced by Bitcoin users. Bitcoin only offers pseudonymity construction by employing public-key cryptography, such that the users do not need to reveal their real identity when creating transactions. However, analyses show that the blockchain data is useful to identify the related parties of standout transactions. [72, 95, 92].

The privacy-preserving feature in Monero, as defined in the original Crypto-Note protocol [108], consists of two parts, namely Linkable Ring Signature (LRS) [64] and one-time public key (OTPK). LRS prevents double-spending in Monero by allowing each key pair to create only one signature. If the same key pair creates more than one signature, the system will be able to detect the occurrence because those signatures will reveal an identical secret key, namely `key image`. LRS provides untraceability of the sender because the real sender is indistinguishable among a set of senders. In a Monero ring signature, the probability P of guessing the real signer is $P = {}^1/_r$, where r is the ring size or the number of ring members.

OTPK is a Diffie-Hellman key exchange protocol, where secret keys are exchanged between two parties using an insecure communication medium [32]. OTPK provides unlikability of the receiver. Multiple transactions are unidentifiable that they belong to the same receiver because each address only appears once. In Monero, the receiver's OTPK is created by the sender by using the receiver's master public key pairs in the form of transaction outputs. We call these outputs as keys to avoid confusion. Each key is associated with an amount of Monero coins (XMR).

We define a transaction T as a function to transform a set of R inputs (we will also call these inputs as rings) into a set of O keys such that $T := R \rightarrow O$ where $R = \{o_i, o_j, ..., o_m\}$ and $O = \{o_n, o_o, ..., o_r\}$. The input R is a ring signature which consists of one or more existing keys, where there is only one key to be spent in the ring, and the other ring members pose as decoys or mixins. The transaction T produces one or more new keys, which contains coins. Future transactions can spend these keys.

7.2.2 Protocols in Monero Network

There are two types of node based on its availability: private node and public node[2]. A private node can only be accessed by the person (or party) that established the node, while a public node (or also called as a public remote node) allows incoming connections from other users. At the time of writing, there are 2,795 Monero nodes[3]. Among the scanned Monero nodes, 66 of them offer RPC connection. All nodes that provide RPC connection are public.

Monero utilises two types of protocol in its network: `peer-to-peer` (P2P) protocol and RPC protocol [16]. Monero's P2P protocol utilises the Levin protocol created by Andrey N. Sabelnikov [77], which was implemented as `epee` library[4]. The A node connects to other nodes through P2P connections that create the whole Monero network. A node receives new updates about the blockchain from other nodes it connects to through P2P connection. Some examples of the updates are new block creation, block reorganisation, and newly created transactions.

On the other hand, a node and a wallet establish an RPC connection. The RPC connection provides a way for the wallet to get information about Monero blockchain without storing the whole blockchain data. Public nodes open the RPC connection to any wallets, while private nodes restrict the RPC connection for private use. Figure 7.2 illustrates the difference between the two communication protocols in Monero.

7.3 Related Work

7.3.1 Address Clustering

In most cryptocurrencies, the address works similar to a bank account, where a payer can use the payee's address as the receiver of the payments. However, unlike bank accounts, there is no relationship between the real identity of the users and the address, as described in Bitcoin whitepaper [81]. Address clustering is one of the methods to identify the relationship among different cryptocurrency addresses. Known Bitcoin analyses used this technique [72, 92]. Change address clustering is one of the methods in address clustering. It is done by

[2]There are different terms to denote different types of Monero nodes, for example, the one provided by https://moneroworld.com/#nodes. However, in this chapter, we simplify the terms by only using private node and public node.

[3]The data was from https://autonode.xmr.pm on 10 May 2019, 4:15 pm AEST.

[4]https://github.com/hyle-team/epee

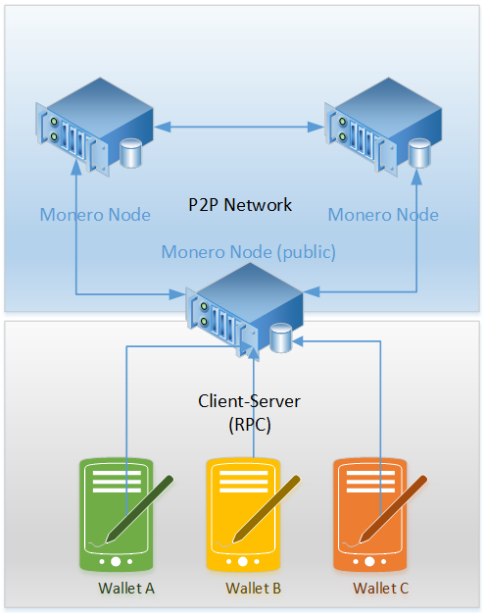

Figure 7.2: Monero network consists of two protocols: P2P network and RPC network.

identifying the change addresses on Bitcoin transactions, then putting the change addresses to the same clusters as the originating (senders') addresses.

7.3.2 Merge Avoidance

Merge avoidance was proposed in 2013 to address privacy leaks in Bitcoin transactions when a user spends a large or unique amount of coins, such that the information can reveal information about related participants (the payer and the payee) [44]. Merge avoidance solves this problem by urging the sender to split a payment into multiple transactions, instead of sending the payment in one transaction. Splitting the payments into smaller amounts reduces privacy leaks.

Merge avoidance is considered an easy solution to the privacy leak. It can also effectively reduce the use of change address. However, merge avoidance depends on the sender, which may not be motivated to create several transactions. Furthermore, merge avoidance can also increase the transaction fees that must be paid by the sender, which hinders its usage.

7.3.3 Cryptocurrency Network Analyses

Network analysis is one of the underexplored areas in cryptocurrency privacy [46]. Cryptocurrency developers usually try to solve the network privacy leak by integrating The Onion Router (Tor) protocol[5] [33], I2P[6], or Kovri[7] to their systems. However, research shows that Tor is not a perfect solution to anonymise network connection [13, 46]. Concerning cryptocurrency, researchers managed to reduce the role of the Tor by employing Bitcoin's blacklisting feature to block network packets from Tor relays [11, 12].

7.4 Security Model

We consider an `Attacker` that operates an honest-but-curious node. A set of wallets connects to an honest-but-curious node through the Internet. The node provides a service to the wallets regarding the blockchain data maintained by the node. The wallets are allowed to send requests to the node, where the node always responses with the correct answers. The `Attacker` can also be the same party that provides the wallet application to users. However, the `Attacker` does not enable the wallet application to send any statistical data related to users' transactions.

The transmitted information in `Attacker`'s node is collected and analysed. The purpose is to guess the spent keys of the transactions known by the node. Although the users connect their wallets through an anonymous network such as Tor or Kovri, the `Attacker` can still conduct the proposed analysis, because the analysis is independent of the IP addresses of those wallets.

If the wallet connects to `Attacker`'s node through an anonymous or a secure network, then the `Attacker` cannot read the network packets. However, the `Attacker` can still get access to the requests and responses information, since the node as one of the related parties always receives unencrypted packets; it can always decrypt the encrypted packets. Therefore, the `Attacker` can still conduct the network analysis even if an anonymous or a secure network is used by the wallet users to connect to the node.

[5] https://www.torproject.org
[6] https://geti2p.net
[7] https://kovri.io

7.5 Guessing Spent Keys

7.5.1 Overview

Guessing spent keys presented in this chapter is a passive attack conducted by a public node as the `Attacker`. The attack targets change coins owned by wallets that connect to the public node, which is operated by an attacker. If the wallets always connect to the same node and they spend the change coins, then the keys related to the change coins appear at least twice in the same node: when the wallet creates the coin and when the wallet spends that coin.

We denote a wallet W and a node N. The wallet W creates a transaction T_i that produces new outputs (keys) o, such that $T_i := R_i \rightarrow \{o_i^0, o_i^1, ..., o_i^m\}$ where $o_i^k \in R_i$. It is also assumed that o_i^k is the change coin that belongs to the sender. Later, the wallet W creates a new transaction T_j that spends o_i^k such that $T_j := R_j \rightarrow \{o_j^0, o_j^1, ..., o_j^m\}$ where $o_i^k \in R_j$ and $i < j$. If the wallet W creates both transactions T_i and T_j while connected to the node N, then the node N knows that the key o_i^k is created by a wallet that connects to it and again the same key is involved either as a mixin or a real spent key in a newer transaction. For each transaction T_j received by node N, the `Attacker` checks whether there exists a previous transaction T_i such that:

1. T_i was sent from the same wallet as T_j, and
2. a key o_i^k of T_i is contained in one of the inputs to T_j.

If both 1 and 2 are satisfied, the `Attacker` concludes that o_i^k is spent in T_j.

7.5.2 Survey on Monero Wallets

We surveyed smartphone-based Monero wallets in iOS and Android. The survey was conducted on 10 June 2019 by installing the Monero wallets on compatible devices and evaluating the settings on the respected wallets. Google Playstore provides the number of downloads of Android applications. However, the Apple Appstore does not provide the same information. Table 7.1 provides the detail of the survey. There were nine surveyed applications from iOS and Android platforms. Seven of the surveyed wallets only support Monero, while the other two wallets support multiple currencies. One wallet asks the users to set up their nodes, while there is only one wallet offers auto-switch node feature with an initial node provided by the wallet.

Table 7.1: Node selection in smartphone-based Monero wallets.

No	Platform	Wallet Name	Wallet Type	Node Selection	Detail	Number of Downloads
1	iOS	Cake Wallet	Single currency	Auto switch node	Eight nodes available Initial node: node.xmr.pt:18081	No data
2	Android	Coinomi	Multi currency	Default	No access to node setting.	500,000+
3	iOS, Android	Edge	Multi currency	Default	No access to node setting.	100,000+ (Android)
4	iOS, Android	Exa Wallet	Single currency	Default	No access to node setting. (monero.exan.tech)	100+ (Android)
5	iOS, Android	Monero Freewallet	Single currency	Default	No access to node setting.	100,000+ (Android)
6	Android	Monerujo	Single currency	No default node	Node setup required	10,000+ (Android)
7	iOS	MyMonero	Single currency	Default	mymonero.com	No data
8	iOS, Android	WooKey Wallet	Single currency	Default	Four nodes available for selection. Default: node.moneroworld.com:18089	100+ (Android)
9	iOS	X Wallet	Single currency	Default	node.moneroworld.com:18089	No data

The result of the survey shows that the majority of smartphone-based Monero wallets only connect to specific nodes. However, it is not a trivial work to determine how many users are prone to the proposed attack due to the lack of an accurate number of downloads, especially for iOS applications. Furthermore, the multi-currency wallet's number of downloads may not precisely reflect the number of wallet usages for managing Monero because the users can download the wallet to manage cryptocurrencies other than Monero. Nevertheless, it is essential to note that most smartphone-based Monero wallets are prone to the proposed passive attack, except for `Cakewallet` which has a feature to switch nodes automatically. However, there is currently no sufficient documentation on the feature's implementation.

7.5.3 Potential Cases

Let there exist four Monero users: Alice, Bob, Charlie, and Dave. Alice sends a payment to Bob in a transaction $T_1 := R_1 \rightarrow \{o_1^0, o_1^1\}$. The payment to Bob is contained in key o_1^1, where she also receives change coins in o_1^0. The transaction T_1 is sent to a public node N. Node N knows that transaction T_1 produces two keys: o_1^0 and o_1^1. All users in our cases use wallets that always connect to the same public nodes. For any user $X \in \{Alice, Bob, Charlie, Dave\}$, we say that transaction $T_X := R_X$ is a transaction created by X. Transaction T_X contains a ring R_X. There are four possible cases of the above example, where one of the cases would cause false positive. The detail is as follows.

1. Alice and Charlie use wallets that connect to the same node, N.

   1A. Alice creates transaction T_A that spends o_1^0 then sends transactions T_A to node N.

   Node N correctly guesses that o_1^0 is a change key and transaction T_A spends o_1^0.

1B. Charlie creates transaction T_C that uses either o_1^0 or o_1^1 as a mixin. Charlie then sends transaction T_C to node N. Alice does not do anything.

According to the rule, node N guesses that transaction T_C spends key o_1^0. We call this a false-positive case.

1C. Alice creates transaction T_A that spends o_1^0. Charlie creates transaction T_C that uses o_1^0 as a mixin. Alice and Charlie send transactions T_A and T_C to node N.

Node N cannot determine whether the key o_1^0 is spent in T_A or T_C. Node N guesses that the key o_1^0 is spent in either transaction T_A or T_C. We call this as an error case.

2. Alice, Bob, and Dave use wallets that connect to different nodes, namely N, M, and Q respectively.

2A. Alice creates transaction T_A that spends o_1^0. Dave creates transaction T_D that uses o_1^0 as a mixin. Alice sends transaction T_A to node N, while Dave sends the transaction T_D to node Q.

Node N receives information about T_D from node Q (or other nodes using P2P connection). Similar to case $1A$, Node N correctly guesses that the key o_1^0 is spent in T_A.

2B. Alice creates transaction T_A that spends o_1^0, while Bob creates transaction T_B that spends o_1^1. Alice sends transaction T_A to node N, while Bob sends transaction T_B to node M.

Node N can guess that key o_1^0 is spent by T_A. However, node N cannot guess the spent keys in T_B although it receives the information about transaction T_B from node M (or other nodes) through P2P connection. In this case, node N does not make any guess with regards to o_1^1.

7.5.4 False Positive Case

We define a false-positive as the event of node N to incorrectly guess that a key o has been spent in a transaction T, while the key o is unspent. We denote O as the set of all keys in the blockchain. We denote O_N as the set of all keys that were created by wallets that connect to the node N, where $O_N \subseteq O$. We also denote O_N^{change} as keys that contains change coins, where $O_N^{change} \subset O_N$. Here, to calculate P_{fp}, we assume an idealised model for how a wallet chooses its set of $r-1$ mixins for each transaction, namely the $r-1$ mixin keys are chosen as a uniformly random set of $r-1$ keys from the set O of all keys in the blockchain. We discuss the validity of this assumption later in the chapter.

We formulate the probability P_{fp} of false positive described as case `1B` in Section 7.5.3 as follows.

$$\begin{aligned} P_{fp} &= 1 - \frac{\binom{O-O_N}{r-1}}{\binom{O}{r-1}} \\ &= 1 - \frac{(O-O_N)!}{O!} \cdot \frac{(r-1)! \cdot (O-(r-1))!}{(r-1)! \cdot (O-O_N-(r-1))!} \\ &\underset{O>>O_N+r}{\simeq} 1 - \frac{1}{O^{O_N}} \cdot (O-(r-1))^{O_N} \\ &= 1 - (1 - \frac{(r-1)}{O})^{O_N} \\ &\underset{r \cdot O_N << O}{\simeq} (r-1) \cdot \frac{O_N}{O} \end{aligned} \tag{7.1}$$

If $(r-1) \cdot O_N$ is much smaller than O, then the probability of false positive P_{fp} is negligible.

We define δ_o as the ratio between O_N and O, where $\delta_o = \frac{O_N}{O}$. To get insights on what information can be deduced by a Monero node when δ_o is low, we created a set of simulations and observed the synthetic data sets. Figure 7.3 shows that the false-positive rates from the experiment are close to the theoretical false positive based on Equation 7.1. When the value of δ_o is around 0.5%, the average false-positive rate is close to 5%. This result shows that if δ_o is low, then P_{fp} is ten times δ_o. When δ_o is around 10%, the P_{fp} is around 65%. The result also shows that the false positive rate grows almost linearly to δ_o.

7.5.5 Error Case

We define the error probability P_{error} as the event of more than one transactions T_i and T_j to include a change key $o_k \in O_N$ in their rings. Transaction T_i spends the key o_k and transaction T_j uses o_k as a mixin. Although the creator of T_i can obviously determine that transaction T_i spends o_k and at the same time knows that o_k in T_j is only a mixin, node N is unable to distinguish the key spender. We modify Equation 7.1 to get the error probability as follows.

$$\begin{aligned} P_{error} &= 1 - \frac{\binom{O-O_N^{change}}{r-1}}{\binom{O}{r-1}} \\ &\underset{r \cdot O_N^{change} << O}{\simeq} (r-1) \cdot \frac{O_N^{change}}{O} \end{aligned} \tag{7.2}$$

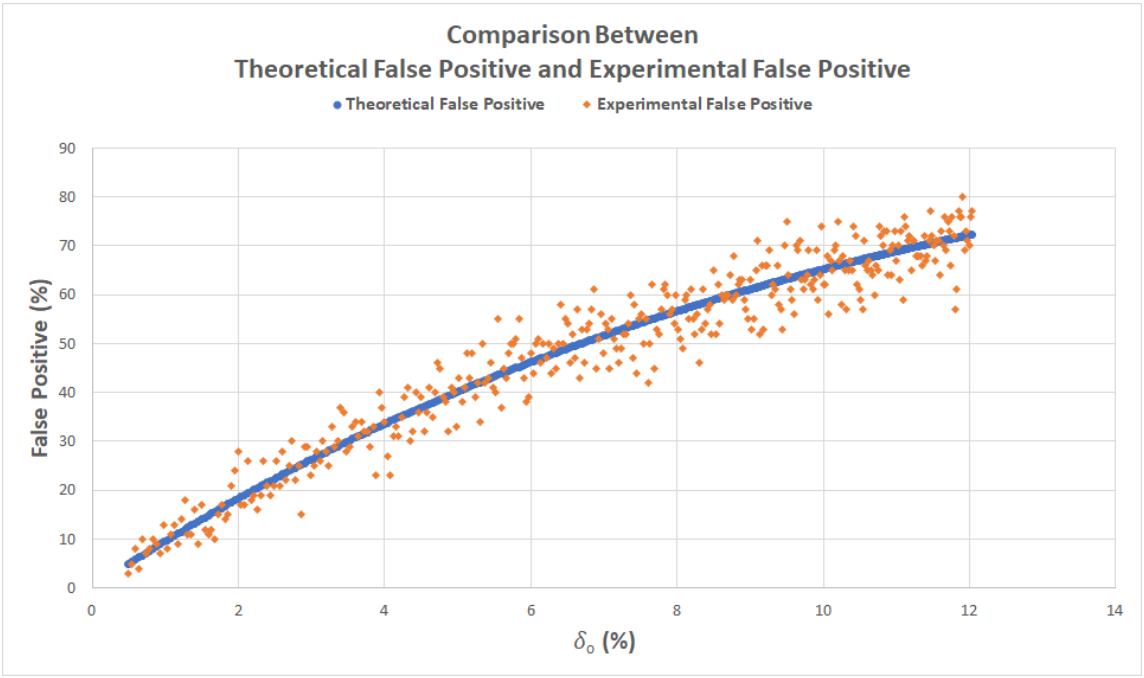

Figure 7.3: Comparison between experimental false positive and theoretical false positive where δ_o is low.

We define $\delta_{o.change}$ as the ratio between O_N^{change} and O, where $\delta_o = \frac{O_N^{change}}{O}$. We also define w as number of wallets that connect to node N.

To further investigate the error rate of our attack, we tested the value of error rate on different $\delta_{o.change}$ and the number of wallets w through simulations. For that purpose, we created four sets of scenarios with the same number of transaction $t = 10,000$ and different number of wallets w which are 100, 1,000, 5,000, and 10,000. For a starter, each wallet only had one key to spend. The result in Figure 7.4 shows that error rates on all scenarios increase as $\delta_{o.change}$ increases.

Our simulation used multiple wallets that connects to the `Attacker`'s node. In the simulation, the `Attacker` managed to guess all (100%) of the spent change keys. However, in addition to the correct guesses, the `Attacker` also made wrong guesses. It is also infeasible to distinguish between correct and wrong guesses. The wrong guesses mean that the number of `Attacker`'s guesses exceeds the number of spent change keys. The ratio between the wrong guesses and the correct guesses is the error rate.

The simulation result also shows that the number of wallets significantly impact the error rate. The error rate for $w = 10,000$ is approximately 50% when $\delta_{o.change} \approx 3.5\%$. The error rate for the same scenario further increases to nearly 90% when $\delta_{o.change} \approx 10\%$. The significant difference on each scenario with different number of wallets is observable when $\delta_{o.change} \approx 5\%$, where the error rate in scenarios with w of 100, 1,000, 5,000, and 10,000 are 34.8%, 36.4%, 49.9%, and 58.9% respectively. However, when $\delta_{o.change} \approx 0.2\%$, the error rate in those scenarios are 2.07%, 2.08%, 3.7%, and 5.3%. The results in scenarios with $w = 100$ and $w = 1,000$ reflect the formula given in Equation 7.2, where the false-positive rate equals to 10 times $\delta_{o.change}$. However, the false-positive rates on simulation scenarios with $w = 5000$ and $w = 10,000$ exceed the theoretical error rate. The result is likely because the average number of transactions per wallet is low. Thus, the number of change keys involved in the transactions is also low. In summary, the Equation 7.2 applies to scenarios where a wallet repeatedly creates transactions, and the transactions reuse change keys in the next transactions.

In the simulation, the number of wallets is proportional to the number of O_N^{change}. However, it is inversely proportional to the number of spent change coins in the transactions. The result in Figure 7.4 indicates that scenarios with more wallets have a higher error rate. The finding is consistent on any given $\delta_{o.change}$.

The formula in Equation 7.1 shows that to get an accurate P_{error}, the O_N^{change} needs to be precisely defined. However, our security model does not allow the `Attacker` to gain extra information from the wallet application other than the transactions that the wallet creates. Therefore, the value of O_N^{change} can only be estimated. In our simulation, we defined that

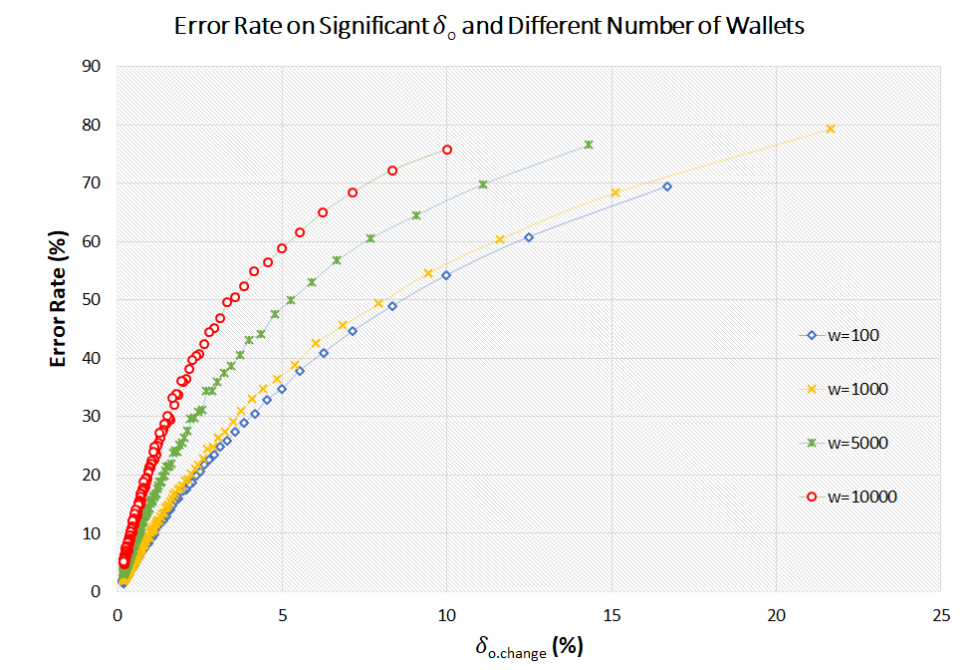

Figure 7.4: The error rates on different $\delta_{o.change}$ values. Number of wallets also impacts the error rate.

each transaction follows the rule $T_x := R_x \rightarrow \{o_x^0, o_x^1\}$, where one of the keys in $\{o_x^0, o_x^1\}$ is the change for the sender. In a general setting where a transaction requires a change, an estimated value of O_N^{change} is equal to the number of transactions.

7.6 Mitigation Strategies

The mitigation strategies presented in this section try to reduce the probability of a node correctly guesses the spent keys of transactions created by the associated wallets. The methods are either by increasing the false-positive rate or by dividing information into several nodes. We propose three mitigation strategies, namely public node centralisation, `Naïve Multinode Wallet (NMW)`, and `Node-Scheduling Wallet (NSW)`.

7.6.1 Public Node Centralisation

Public node centralisation allows only one available public node for all thin wallets to connect to Monero network. The purpose of centralising public node is to increase the false-positive rate of node N when guessing the spent key. By forcing all thin wallets to connect to node N, the number of keys O_N may also increase. As a result, the value of δ_o increases. However, since it is difficult to determine the exact number of transactions created by the thin wallets, it is also challenging to determine the effectiveness of the mitigation strategy. Furthermore, the decentralised nature of Monero also makes the strategy infeasible to implement.

7.6.2 Naïve Multinode Wallet

To reduce the probability of a node successfully guessing the real spent keys which previously appeared in the network, we propose a naïve solution by connecting to multiple nodes instead of connecting to only one node as in a traditional Monero wallet. We call this solution `Naïve Multinode Wallet (NMW)`. We denote n as the number of nodes that a wallet connects to during the wallet's lifetime. We also define a success rate as a ratio between the correct guesses and the total number of change keys that a wallet spends. `NMW` reduces the success rate by dividing transaction-related information to randomly selected nodes.

We conducted experiments to evaluate the effectiveness of the proposed `NMW` with various number of nodes. The experiments show that the success rate decreases when the number

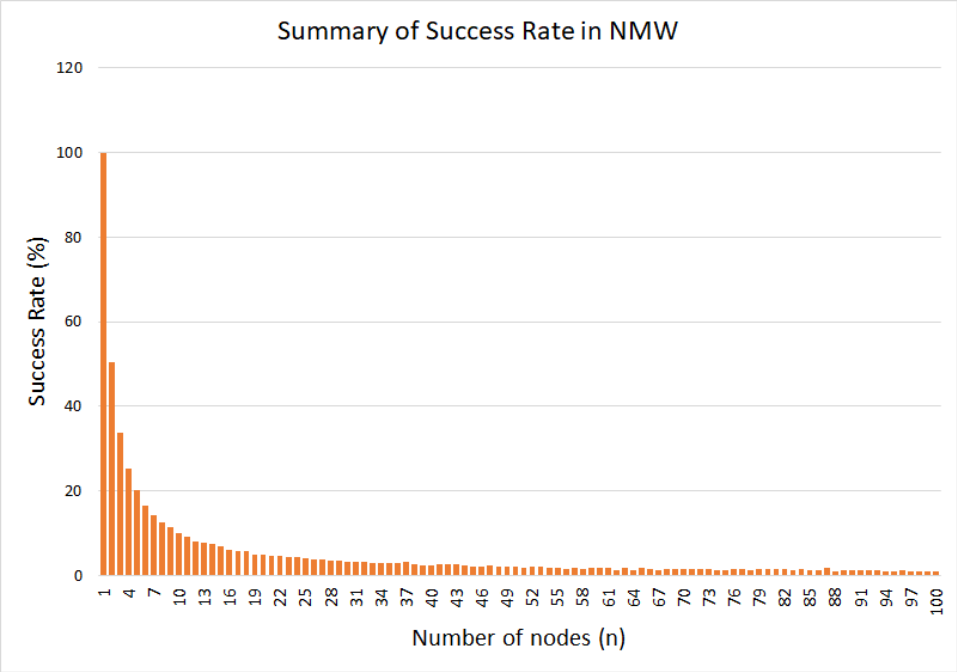

Figure 7.5: The average error rate of NMW using different number of nodes.

of nodes n increases. The average success rate for $n = 2$ is 50.3%. For $n = 5$ and $n = 10$, the average success rates are 20% and 10% respectively. The result in Figure 7.5 shows that the success rate decreases for each additional node. It also shows that it requires more than 50 nodes to reduce the success rate to 2%. However, even if 99 nodes are involved in a transaction creation, the success rate for each node achieves up to 1%.

7.6.3 Node-Scheduling Wallet

We assume that every node has a unique identifier which we call as `NodeID`. The unique identifier `NodeID` can be in the form of an IP address or a domain name. We also denote an `own key` as a key that contains a change coin and can be spent by using its private key. The `own key` can be spent in future transactions. To improve the usability of the nodes, we propose a `Node-Scheduling Wallet (NSW)`. In the `NSW` solution, the wallet identifies the `NodeID` of where each transaction was sent to. Then, the wallet records the `own key-NodeID` tuple.

The minimum number of nodes required for the node-scheduling wallet in a transaction

depends on the number of distinct NodeID of the inputs created by the node, such that $n_r = n_d + 1$ and $n_d \leq n_i$ where n_r is the required number of nodes, n_d is the number of distinct NodeID of the transaction's inputs, and n_i is the number of the transaction's input. For example, if the transaction contains only one input, then the minimum number of nodes is two. NSW is more efficient compared to NMW presented in Section 7.6.2 in terms of number of nodes, where NSW only requires a small overhead of storing the NodeID for each created transaction.

We simulated the proposed NSW solution. In our simulation, each transaction only contains one input; therefore, the maximum number of required nodes to connect is two. Compared to NMW with 100 nodes, NSW is more efficient because it uses less nodes when $n_d + 1 < 100$. In Monero mainnet, the average number of inputs $n_i = 5.6$, therefore NSW requires $n_r = 5.6 + 1 = 7.6 \approx 8$ nodes when $nd = n_i$. NSW also reduces the success rate to 0% by utilising eight nodes, compared to 12.6% success rate in NMW with the same number of nodes. In summary, NSW can fully mitigate the passive attack by wallet service providers. Our result also shows that the false positive rate of NSW is similar to NMW.

7.7 Limitation

Both NMW and NSW solve the problem by distributing information to multiple nodes. However, there is no guarantee that the nodes do not collude by sharing information between themselves, nor whether they operate under the same Attacker that conducts the passive attack. Assuming that the wallet users do not have any information about the colluding nodes and avoid connecting to any of those nodes, then none of our solutions can fully mitigate the attack.

7.8 Conclusion and Future Work

In this chapter, we demonstrate how a passive Attacker can guess the spent change keys of the wallets connected to Attacker's node. The spent keys have appeared as outputs in previous transactions created by the wallets while connected to the same node. The false-positive rate of the guess correlates to the ratio of $\delta_o = \frac{O_N}{O}$. We also investigate the error rate when the Attacker discovers that a key appears in multiple transactions. The number of wallets w that connect to the Attacker's node N impacts the number of spent change keys, which also impacts the error rate.

The majority of the smartphone-based Monero wallets we surveyed provide limited options of Monero nodes to connect. This configuration exposes the wallets to the anonymity problem we present. We also compare two potential solutions, namely `Naïve Multinode Wallet` (`NMW`) and `Node-Scheduling Wallet` (`NSW`) to mitigate the problem. `NMW` requires more than 50 nodes to reduce the guessed spent keys to reducing the success rate to 2% of the total spent keys. On the other hand, `NSW` requires $n_i + 1$ nodes to reduce the success rate to 0%, where n_i is the number of the transaction's inputs.

For future work, we recommend a redesign of the communication protocol between the Monero wallet and Monero node. The new protocol aims to reduce the information received by the nodes. The information received by a node should be insufficient to deduce any information about the spent keys. We also plan to observe different implementations of Monero wallet and how they might leak users' privacy when communicating with Monero nodes. Multi-currency wallets can be useful for users who own multiple cryptocurrencies. However, the implementation of the wallets might compromise the anonymity of the users while creating transactions of different cryptocurrencies.

7.9 Chapter Summary

In this chapter, we present our research on anonymity risks related to Monero wallet service providers, as the first part of our last research question, `Q3`, about Monero third-party services. We identify that the service providers can conduct traceability analysis, especially on change coin spending, after collecting transaction data from their clients' wallets. We propose mitigation strategies to prevent the identified risks. Our proposals distributes a wallet's transaction data to multiple wallets to reduce the wallet service providers' ability to gather transaction data. The solutions show significant decline of traceable inputs.

In the next chapter, we present the second part of our third research question, `Q3`, that investigates mining pools' data publication that leaks transaction anonymity.

Chapter 8

Monero Mining Pools Anonymity Leaks

Monero is a cryptocurrency that provides anonymity by default to its users. Monero transactions are infeasible to trace without additional information. Monero utilises Proof-of-Work (PoW) as its consensus method. Miners contribute their computing power in exchange for a financial reward. The miners can opt to join mining pools to reduce their income variance. Mining pools coordinate the miners' mining power in order to generate more reward and distribute the reward based on each miner's contribution to getting the reward. Some mining pools publish the list of won blocks and payout transactions to the miners to provide business transparency.

In this chapter, we present our research on mining pools. This chapter is the second part of our last research question, `Q3`, that explores essential third-party services in Monero ecosystem. We propose analyses of mining pools' data. To achieve the goal, we collected published data from ten mining pools' websites and conduct traceability analyses on the data. We discovered that 59.2% inputs of all Monero transaction inputs we recorded are traceable. We also discovered that the age of the spent coins is between 2.5 hours and 3.3 days.

While we propose methods to improve the accountability of the published information, it is also questionable that the mining pools keep publishing such information which reduces the anonymity of their transactions. Our investigation shows that there is no relationship between publishing mining-related information and the success of a mining pool. Our manuscript, *Transparency or Anonymity Leak: Monero Mining Pools Data Publication*, becomes the basis of this chapter.

8.1 Introduction

At the time of writing, Monero is in the top 15 of the most valuable cryptocurrencies list. Its total market value is US$851million, where each coin is worth US$47.40[1]. Monero is also the most successful CryptoNote-based cryptocurrencies, among other products such as ByteCoin, Boolberry, and Aeon[2]. Nicolas van Saberhagen[3] published CryptoNote protocol in 2013.

Monero utilises Proof-of-Work (PoW) as its consensus method. The memory-bound consensus algorithm called CryptoNight aims to decentralise mining activities, where small miners can mine Monero by using commercial CPUs and GPUs. Similar to Bitcoin and other cryptocurrencies, there are mining pools that coordinate mining powers of multiple miners to increase their cumulative chances of winning blocks.

The current Monero mining pool lacks transparency due to the default anonymity features in Monero transactions. A blockchain observer cannot determine how many blocks each mining pool wins at any period. Therefore the miners are unable to evaluate the fairness of the incentives from the computation shares they submit to the mining pools. In order to improve the transparency, mining pools utilise their websites to publish information related to their mining productions, such as the blocks won and payouts to miners. Although the published information might be useful for miners to decide which mining pools they want to join, the same information is vulnerable to anonymity leak.

Contributions. In this chapter, our contributions are as follows.

1. We identify an anonymity leak problem within the mining pools' published information. We consider two possible cases of traceable inputs in a mining pool M, namely `Case A` and `Case B`, as shown in Figure 8.1.

    - In `Case A`, mining pool M spends mining reward co_1 of won block B_{1000} as an input to transaction t_1. Since we know that mining pool M creates block B_{1000} and transaction t_1, there is a high probability that the input i_{11} spends co_1, and therefore input i_{11} is traceable.
    - In `Case B`, mining pool M creates t_3 as a payout to a miner and receives change coins in either o_{31} or o_{32}. Later, mining pool M creates a new payout t_4 to another miner. The transaction t_4 contains o_{32} in its input i_{41}. In this case,

[1]Data taken from Coinmarketcap on 25 November 2019
[2]https://cryptonote.org/coins
[3]Nicolas van Saberhagen is a pseudonym.

there is a high chance that the mining pool M spends its change coins o_{32} in t_4, and the input i_{41} is considered as traceable.

2. We explore anonymity leak from mining pools' published information. We collected information from ten mining pools and conducted traceability analyses to the data set. We discovered 218,957 traceable inputs. This result is the second largest anonymity leak after Monero implements RingCT feature. Also, our second-order traceability can identify 5,501 traceable inputs.

3. We propose possible countermeasures to the identified problem. Our solutions include data removal and transaction obfuscation as potential countermeasures to prevent future anonymity leak from mining pools' published information.

4. To improve the mining pools' accountability on the produced blocks, mining pools can publish their tracking keys such that observers can calculate the actual block production rate on all mining pools. We also propose the adoption of Slushpool's Hash Rate Proof (HRP) to improve the verifiability of mining information.

5. We analyse the top ten Monero mining pools and extract their features. We identify that the majority of the mining pools publish their mining-related information on their websites. Two major mining pools, *Minexmr* and *Supportxmr*, specialise in Monero mining, while *F2pool* as the third-largest Monero mining pool is a multi-coin mining pool. There is no specific feature that distinguishes the top three mining pools from the rest of the mining pools, which indicates that removing the mining-related information from the mining pools' websites might not alter their popularity.

8.2 Background

8.2.1 Monero Anonymity

Monero implements privacy-preserving cryptographic techniques to achieve sender's untraceability and receiver's unlinkability. The sender's untraceability means that the actual sender in a transaction is infeasible to guess over a set of senders. The receiver's unlinkability means that it is challenging to determine whether multiple transactions pay to the same receiver.

Linkable Ring Signature (LRS) [64] and **one-time public key (OTPK)** were implemented to support the sender's untraceability and receiver's unlinkability subsequently

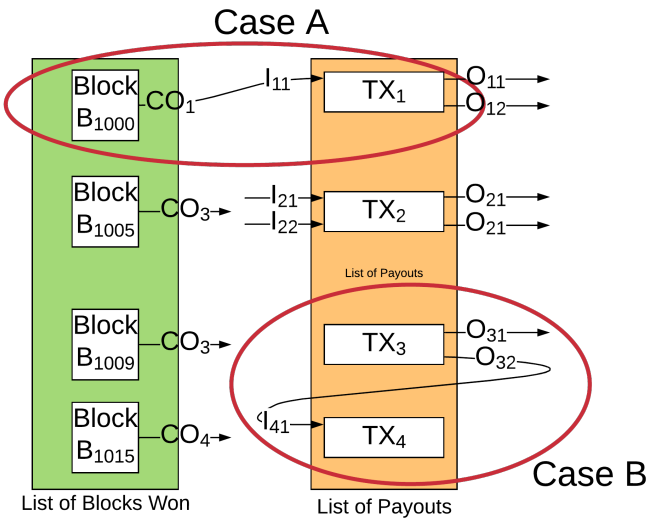

Figure 8.1: Possible cases of traceable inputs from mining pools' published information.

[108]. Monero's **LRS** obfuscates the real spent coins among other coins that pose as decoys or mixins. A sender selects the mixins from outputs of recorded transactions in the blockchain which have the same amount as the spent coins.

Confidential Transaction [70] was integrated to Monero's ring signature to create RingCT [83]. RingCT hides the number of coins transacted while at the same time makes sure that the sender does not create new coins. RingCT was later improved with Bulletproofs [18] to reduce the signature size of Monero transaction. The Confidential Transaction enables the users to select mixins from any compatible transactions with any coin amount. Therefore the pool of potential mixins is much greater than before the implementation of Confidential Transaction in Monero.

A Monero transaction `T` consists of a set of inputs `I` and a set of outputs `O`. The transaction `T` spends the inputs `R` and produces the outputs `O` such that $T := I \rightarrow O$. An input in a Monero transaction consists of a ring `R`. The members of the ring `R` are outputs from existing transactions stored in Monero blockchain. For each ring, there is only one real member to spend (the real spent key), where the other ring members are decoys (mixins), such that the real spent coin is indistinguishable among the ring members.

8.2.2 Monero Consensus, Mining Activities, and Mining Pools

Monero implements Proof-of-Work (PoW) as its consensus method [108]. PoW utilises computing power to determine which party can extend the blockchain. Any parties that dedicate computing power for PoW purposes are called miners. On each cycle, miners compete with each other to solve a computing problem, where the miner that proposes the best solution will add a new block to the blockchain.

There are generally two ways for a miner to mine cryptocurrencies: by mining individually (also called as solo mining) or by joining a mining pool (also called as pool mining). A mining pool is a mining service that allows miners to share the mining workload. The miners that join a mining pool expect to stabilise their income from the mining activities [37]. Mining pools pay the miners according to the miners' contributions based on the agreed schemes. The mining pool operator also takes a fraction of the mining profits as a reward for the mining pool service.

8.2.3 Research on Monero Mining Activity

Monero mining activity is sometimes associated with crypto-jacking [36, 87]. Monero's memory-bound mining algorithm can run on commercial CPUs. Therefore specific hardware such as ASIC machine is not required. Coinhive[4] and other similar services enable website owners to embed mining scripts inside their webpages, such that the scripts will run on visitors' machine and bring profits to the website owners.

A group of researchers also quantified mining activity by collecting information about created blocks and payouts made by 18 mining pools [79]. They extracted 73,667 payouts and 210,800 blocks from those mining pools. Based on the captured information, they estimated that around 0.438 transactions on each Monero block are related to mining activities. Furthermore, they discovered that the number of mining-related transactions correlates with Monero market price [79].

[4]Coinhive dissolved on March 2019.

8.3 Analyses on Mining Pool-related Information

8.3.1 Overview

We investigate the impact of mining-related data publications by mining pools towards transaction traceability. We assume that mining pools spend both their block reward and change coins to pay their miners that have contributed to the mining pools.

The steps of our analyses are as follows. First, we surveyed mining pools. We discovered ten mining pools published information related to Monero mining activities. We then collected the data from the ten mining pools as our primary data set. Then, we analysed the data set to identify the traceability of the transactions. We defined possible scenarios for these transactions and then adopted known traceability analyses to our data set. We further explored our data set to discover known Monero addresses' activities, the use of unencrypted Payment ID (UPID), spent outputs age, and how the traceable inputs impact other transactions' anonymity.

8.3.2 Public Information from Mining Pools

Table 8.1 shows our survey on ten mining pools regarding information they publish on their websites without the need of creating accounts on the mining pools' systems. The `Block Won` column shows that all mining pools publish blocks they produced. `Global Payout Data` indicates whether a mining pool publishes a list of all payouts they made to the miners. Our survey data shows that only two mining pools, *Xmr.nanopool* and *Minexmr*, do not publish payouts data. `Individual Payout Data` shows whether a mining pool provides a list of payouts for user-supplied addresses without logging in. The survey data indicates that although *Xmr.nanopool* and *Minexmr* do not publish all payouts data, they still provide individual payout data. It means that the payout data is available as long as the requester knows the miners' Monero addresses.

`Bulk Payout` indicates whether a mining pool sends payments to miners in bulk (many recipients in one transaction), whereas `Individual Payout` indicates whether a mining pool sends payment to miners on different transactions. Our survey shows that only *Xmr.nanopool* only allows bulk payouts, whereas *2miners* only supports individual payouts.

Table 8.1: The mining pool survey result. Data marked with asterisk (*) means there is no backdate data search. N/A means we could not find any data samples.

No	Pool Name	Block Won	Global Payout Data	Individual Payout Data	Bulk Payout	Individual Payout	Miners' Address
1	supportxmr.com	Yes	Yes	Yes	Yes	Yes	No
2	xmr.nanopool.org	Yes	No	Yes	Yes	No	Block Finder
3	minexmr.com	Yes	No	Yes	N/A	N/A	N/A
4	dwarfpool.com	Yes*	Yes*	No	Yes	Yes	No
5	monerohash.com	Yes*	Yes*	No	Yes	Yes	No
6	crypto-pool.fr	Yes	Yes	No	Yes	Yes	No
7	xmrpool.net	Yes	Yes	Yes	Yes	Yes	No
8	xmrpool.eu	Yes	Yes	No	Yes	Yes	No
9	2miners.com	Yes	Yes*	No	No	Yes	No
10	coinfoundry.com	Yes	Yes	Yes	Yes	Yes	Block Finder

8.3.3 Data Collection

We collected information from ten active mining pools. These mining pools provide various information, such as the list of blocks they won, payouts, even miners' addresses that created the blocks they won. We identified 272,414 blocks B and 157,931 payouts (transactions) T created by the ten mining pools[5]. The transactions also contain 373,559 inputs I with 2,603,108 mixins M. The blocks B contain coinbase transactions that produce new outputs, CO. The number of CO is identical to the number of new blocks B. Transactions I contain a total of 1,181,501 outputs O.

Figure 8.2 shows the distribution of the number of transactions we recorded. *Supportxmr* has the most significant percentage, with 82.4% of all payouts, followed by *Crypto-pool* which contributed 13.7% of all payouts. *Xmrpool.eu* and *Xmrpool.net* are accounted for 2.1% and 1% of all payouts, respectively. Not every mining pool provides the same information and in the same fashion. As a result, we could not find any payouts made by *Minexmr*, as shown in Table 8.1.

Table 8.2 shows the detail of the payouts. From the data we collected, there are a total of 851,341.8 XMR, or about US$54.49million[6], paid by the mining pools to the miners. The information also shows that although *Crypto-pool* has a much lower number of payouts, its total payouts is 50% bigger than *Supportxmr*. There are mining pools that provide the number of payees for each payout. However, there are also mining pools that do not publish the number of payees information. We adjusted the empty information by using estimation based on the number of outputs on the payouts. In average, *Crypto-pool* pays 2.54 XMR to each payee. *Monerohash* pays an average of 1.13 XMR to each payee, followed

[5]As of 14 November 2019.

[6]US$64 per XMR, according to Coingecko.com exchange rate as of 14 November 2019.

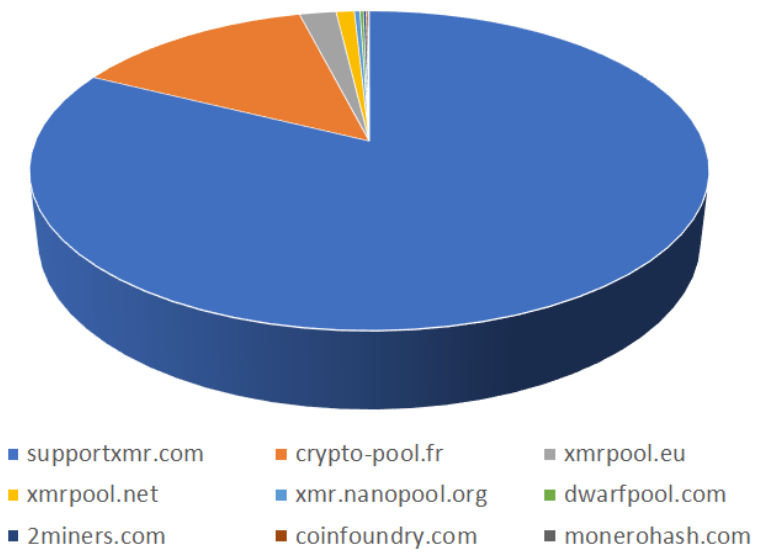

Figure 8.2: The distribution of the captured mining pools payouts.

by *XmrpoolNet* and *Supportxmr*.

8.3.4 Traceability Analysis

We base our traceability analysis on mining pool business processes. Mining pools can use their mining reward to pay their miners. It is also possible that the payouts to the miners spend change coins from previous payouts. Based on these two possibilities, we constructed our data set by using the following rules $R1$.

$R1.1$ The transaction t_1^i must be from the set of transactions T

$R1.1$ Each mixin $m_1^i \in M$ in the transaction inputs is either:

(a) a coinbase transaction output where the same mining pool wins the block, or

(b) an output of a transaction created by the same mining pool.

We apply the rules $R1$ to our data set T, M, and I that we curated from the ten mining pools to produce T', I', and M'.

Table 8.2: The details of payouts data from ten mining pools.

Mining Pools	Total XMR	Num.Payouts	Avg. XMR/Payouts	Total Payees	Avg. XMR/Payee
Crypto-pool.fr	497,377.91	21,565	23.06	195,483	2.54
Supportxmr.com	333,155.12	130,097	2.56	724,373	0.46
Xmrpool.eu	13,439.87	3,292	4.08	41,442	0.32
Xmrpool.net	6,292.01	1,638	3.84	7,690	0.82
Monerohash.com	928.84	106	8.76	822	1.13
2miners.com	118.06	259	0.46	584	0.20
Coinfoundry.com	30.00	170	0.18	1,692	0.02
Xmr.nanopool.org	-	500	-	3,949	-
Dwarfpool.com	-	304	-	517	-

By using the rules, we identified 147,827 transactions T' (where $T' \subset T$) with 275,499 inputs I' (where $I' \subset I$) that contain qualified 393,525 mixins M' (where $M' \subset M$).

Next, we conducted traceability analysis by using the following rules $R2$.

$R2.1$ Each input $i_2^i \in I'$ contains exactly one mixin $m_2^i \in M'$, where the mixin m_2^i is from the same mining pool as i_2^i.

$R2.2$ For an input $i_2^i \in I'$ that contains x number of mixins from the same mining pool, rule $R2.1$ must identify all $x - 1$ mixins.

By applying rules $R2$ to I', we identified 218,957 traceable inputs I'', where $I'' \subset I'$. The traceable inputs are 59.2% of all inputs in I and 78.7% of all inputs in I'. The result is roughly 3 to 4 times more successful compared to key reuse traceability analysis after Monero hard forks. [48, 118]. However, the number of traceable inputs from our proposed analysis is less than the known zero mixin and cascade effect, ring usage, and Monero unencrypted Payment ID analyses. The comparison is shown in Table 8.3.

8.3.5 Multi-Candidate Untraceable Inputs

We also discovered that our rules in Section 8.3.4 were unable to trace 56,842 inputs from our data set I''. There are 586 inputs contain one ring member that fit in our rules R1 and R2. However, other inputs have spent the associated ring members of these inputs. It is possible because of our incomplete data set. Mining pools' transactions that are not payouts, such as cashing out their profits are not published. Table 8.4 shows the untraceable inputs detail.

Table 8.3: The result of traceable inputs from known traceability analysis techniques.

Traceability Techniques	Traceable Inputs
Zero mixin and cascade effect [79]	3,937,331
Ring usage [117]	1,142,383
Monero unencrypted Payment ID [117]	332,987
Mining pool payouts (our proposal)	**218,957**
Key reuse [48]	73,321
Key reuse [119]	53,162
Closed set [125]	3,017

Table 8.4: Multi-candidate untraceable inputs in our data set.

Num. of Qualified Ring Members	Data Count	Percentage
1	586	1.03
2	39,418	69.35
3	13,576	23.88
4	2,860	5.03
5	369	0.65
6	31	0.05
7	2	0.004

Almost all of the untraceable inputs have two or more qualified ring members that fit our rules R1 and R2. These inputs are untraceable because the rules are unable to decide which ring members are spent by the inputs. About 93% of the untraceable inputs are from *Supportxmr*, while 7% of them are from *Crypto-pool*. This finding shows that the mining pools tend to add their outputs as ring members when creating transactions. However, according to the default Monero wallet's mixin selection distribution algorithm, 50% of all mixins must come from blocks generated in the last 1.8 days. It means large mining pools have a higher probability of inserting its outputs as mixins in new transactions because they frequently send payouts to miners compared to small mining pools that send payouts less frequently.

8.3.6 Second-order Traceability Analysis

We define second-order traceability analysis by searching for unlisted transactions $X \not\subset T$. We also define a transaction $t_x := \{i_x^1, i_x^2\} \rightarrow \{o_x^1, o_x^2\}$ where $t_x \in X$, i_x is an input to t_x and

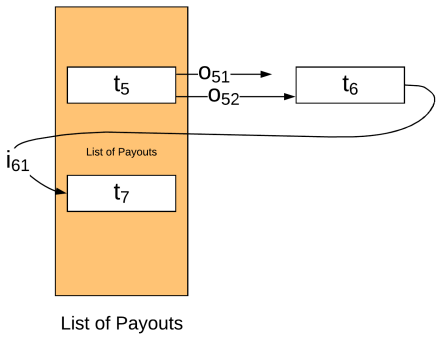

Figure 8.3: Second-order traceability analysis to find $t_6 \in X$ and trace i_{61}.

o_x is an output of t_x. The transaction t_x includes at least one output $o \in O$ in one of its mixins M_X, and at least one of its outputs o_x^m are in M. The case diagram for second-order traceability analysis is shown in Figure 8.3. The search back-tracks all known transactions T to find transactions X. It exhausts all possible paths starting from the outputs to inputs and all mixins. We define IX as all possible traceable inputs from X.

We identified 18,171 possible traceable inputs (I_X) from 11,615 transactions X. However, inputs in I_X has two or more possible spent outputs. We define I'_X as a set of inputs $I'_X \in I_X$ that has exactly one possible spent output that satisfies $o \in O$ and $o \in M_X$. We discovered 5,501 traceable inputs that satisfy the requirements.

8.3.7 Additional Analyses on Mining Pools' Public Information

Although Monero addresses are not usually made public, *Xmr.Nanopool* and *Coinfoundry* publish Monero addresses of miners who won the blocks for the two mining pools. We discovered 339 Monero addresses from our data set[7]. We also utilised the Monero address list published by Palo Alto Networks (PAN)[8]. PAN investigated cryptocurrency mining activities ran by malware applications and discovered 2,341 Monero addresses [43].

We discovered 914 traceable inputs that spent outputs from previous transactions, where the spending transactions and the previous transactions have the same unencrypted Payment ID (UPID). The result shows that the method proposed by [117] is still applicable. However, due to the limited amount of data we collected, we could not produce the same

[7]https://github.com/sonicskye/monero-miningpool-data/blob/master/miner-addresses
[8]https://github.com/pan-unit42/iocs/blob/master/cryptocurrency_miners/xmr_wallets.txt

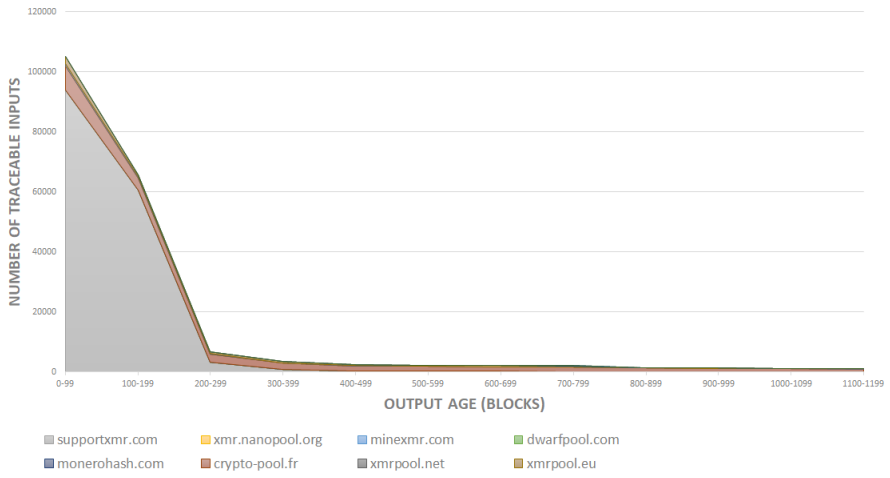

Figure 8.4: The output age of known traceable inputs.

results. The UPID feature might not be available in the future. Monero developers have stated that they will phase out the UPID and replace it with the encrypted Payment ID (EPID) and subaddress[9].

Based on the traceable inputs we discovered in Section 8.3.4, we analysed the age of all identified spent outputs. Monero's current block production time is roughly one block every two minutes[10]. The finding in Figure 8.4 shows that the age of around 81% of the spent outputs is less than 300 blocks or roughly 2.5 hours. The age of the other 18% of the known spent outputs spans from 300 blocks to 9,400 blocks, or between 2.5 hours to 3.3 days.

We also discovered 99,389 spent coinbase outputs. The age of the spent outputs from coinbase transactions also show an identical figure to Figure 8.4. The figure shows that the mining pools do not distinguish whether the spent outputs are from the previous transactions or the coinbase transactions. Although the coinbase outputs cannot be spent at least two hours (or 60 blocks), this restriction does not make much difference to the figure.

From the traceable inputs we discovered in Section 8.3.4, we identified 1,578,094 inputs from 1,057,833 different transactions that utilised the traceable inputs as mixins. About

[9]https://web.getmonero.org/2019/06/04/Long-Payment-ID-Deprecation.html

[10]According to https://bitinfocharts.com/comparison/monero-confirmationtime.html, the current block production time was shifted from one block every minute to one block every two minutes end of March 2016.

98% of the inputs only use less than three of the traceable inputs, where on average, the transactions suffer 12% reduced anonymity or only 78% effective anonymity.

8.4 Possible Countermeasures

We propose possible countermeasures to the identified problem of a privacy leak from the published information by Monero mining pools. Our proposals include data clean-up, transaction obfuscation, and provable data publication.

One of the most obvious ways to avoid the identified problem of traceable inputs of mining pools' payouts is not to make the payout data available for the public. Instead, related parties such as miners can receive limited information from the mining pools. To access payout data, a miner needs to satisfy a simple authentication mechanism, for example, by using his or her Monero address.

8.4.1 Transaction Obfuscation

8.4.1.1 Churning

Mining pools can reduce the traceability risk by churning their coins before spending them to pay their miners. Churning [58] is done by spending the coins and send them to own address(es). The purpose of churning is to increase the total number of mixins or possible spent inputs and therefore improve the sender's untraceability [58]. Churning also increases the distance between the original outputs and the current outputs in the wallet [53].

Churning can mitigate against second-order traceability analysis (Section 8.3.6). Churning increases the difficulty of finding a mining pool's unlisted transactions. With churning, the number of searches s increase by the following formula: $s = (i * r)^c$, where i is the average number of inputs per transactions, r is the average ring size, and c is the number of churning rounds.

8.4.1.2 Own Outputs as Mixins

The analysis we conducted in Chapter 8.3.5 provides an insight into a new obfuscation method creation. If the mining pools keep publishing blocks they won and payout transactions, then their mixin selection algorithm can be improved. We suggest adding at least one

of their outputs as mixins in the newly created transactions. *Supportxmr* uses this scheme. However, it is also important not to create a closed set when creating mixins of own outputs [125]. In order to use the method effectively, a mining pool needs to create more frequent transactions. Setting a lower minimum payout and enabling individual payout can help to increase the number of transactions by the mining pool.

8.4.2 Accountable Data Publication

8.4.2.1 Mining Pools' Monero Tracking Keys

Monero provides an accountability feature called tracking key [108]. Monero uses two sets of private and public keys, $\{(a, A), (b, B)\}$ in its system. A monero address consists of the two public keys $PK = \{A, B\}$, while a Monero private spend key consists of the two private keys $PR = \{a, b\}$. Monero's tracking key consists of one of the public keys and one of the private keys $TR = \{a, B\}$. While the private spend key PR holder can spend coins, a Monero tracking key is ineffective to spend coins. It can only detect incoming Monero transactions.

The tracking key of each mining pool can then be published. The tracking keys of all mining pools can identify which blocks are won by each mining pool by using the following logic. If a mining pool's Monero address receives a payment from a coinbase transaction of a block that consists of mining reward and transaction fees, then it indicates that the mining pool won the block.

Publishing Monero tracking key can weakly mitigate a malicious mining pool that falsifies block production data (mining power or won blocks). However, tracking key publication is voluntary. While most mining pool operators are approachable, it is infeasible to enforce the same policy to solo miners. Unwilling miners or mining pools can also avoid tracking by creating new addresses or subaddresses for every mining cycle.

8.4.2.2 Slushpool's Hash Rate Proof

Slushpool, a Bitcoin and Zcash mining pool, created Hash Rate Proof (HRP)[11], an algorithm to check the correctness of the mining pool's total hash rate. The mining pool also publishes shares data of its miners per hour as the input for its algorithm. Each share data contains three parts:

[11] https://slushpool.com/help/hashrate-proof

- block header (80 bytes)
- coinbase transaction
- Merkle branch

The following method verifies the share data. First, the verifier computes Merkle root hash by using the coinbase transaction and Merkle branch hashes. Then, the Merkle root hash needs to be included in the block header's byte number 36 to 68. Each share also contains a unique string, `'/slush/'`, in its coinbase transaction which verifies that the share was submitted to *Slushpool*.

A Monero mining pool can adopt HRP to improve their mining activities transparency. The benefit is three-fold:

1. to help identify the mining pool that wins a block; the verifier can further use the mining pool's tracking key TR (Section 8.4.2.1) to check whether the mining pool receives the block's mining reward;
2. to prove that a miner wins the block without publishing the miner's address;
3. to prove that a miner's shares contribute to mining the block.

A Monero miner can refer to HRP data to determine whether the payouts she receives from the mining pool reflect her contribution to the mining pool according to the mining pool's reward scheme. The mining pool only needs to publish the total amount of payouts without further details on the payouts, such as transaction IDs, amounts, and recipients' addresses. Although HRP can mitigate incorrect block production data, its problems are also identical to publishing tracking key.

8.5 Mining Pool Feature Analysis

In this section, we discuss possible reasons for publishing mining pool data. We also analyse the characteristics of the top ten Monero mining pools to identify the miners' preferences in mining pool selection.

8.5.1 Mining Pool Data Publication and The Lack of Verification

The root cause for traceable payout transactions is because the mining pools publish the list of blocks they won and the list of payout transactions to the miners. The data published by mining pools is probably for promoting their services. The mining pools would also want to be preferable to miners by providing evidence to the miners that they are productive, robust, and behave honestly [6].

If mining pools do not publish the related mining data, it is infeasible for the miners to determine how many blocks the mining pools won and how many payouts are made by the mining pools to each miner. The Monero mining-related data collection approach is different compared to Bitcoin. In Bitcoin, the mining pools that created the blocks can be determined based on the coinbase transactions' destination addresses which belong to the mining pools or the miners. However, since Monero uses the one-time public key (**OTPK**) technique, the coin receivers are unlinkable.

In Monero, publishing mining information is arguably useful. There is no means to verify the correctness of the published list of won blocks and a list of payouts. Mining pool information portals such as *Miningpoolstats.stream*[12] rely on mining pools' published information on the mining pools' websites. A malicious mining pool can create their fake list of won blocks that consists of blocks that they did not win. An artificial list of payouts can also be created maliciously by creating Monero transactions that pay to their addresses. Monero anonymity hinders the verification of the payouts and the number of coins on each payout.

A malicious mining pool can also reduce their accumulated mining power by modifying their published mining information. The malicious effort can be identifiable by comparing the Monero mining difficulty level and the accumulated mining power from all known mining pools. However, this method is ineffective to identify the malicious mining pool and infeasible to determine how much mining power reduction, since there is a possibility of solo miners that contribute their mining power to Monero.

[12] https://miningpoolstats.stream/monero

8.5.2 Characteristics of Top Ten Monero Mining Pools and Miners' Preference

8.5.2.1 Accumulated Mining Power and Mining Pool Data Publication

Table 8.5 shows several features of the top ten Monero mining pools according to *Miningpoolstats.stream*. *Minexmr* and *Supportxmr*, the top two on the list, are known to publish their won blocks on their respective website. Both mining pools control more than 65% of the total hashrate in the network. However, *F2pool* on the top three of the list does not publish mining-related data such as won blocks and payout transactions on its website except its accumulated mining power.

Table 8.5 also shows the reward systems adopted by the top ten Monero mining pools. The majority of the mining pools use PPLNS as their reward system. *F2pool* is the only PPS-based mining pool. Although *F2pool* charges more pool fee compared to other mining pools, it accumulated enough mining power to sit in the top three of the list. The result attests that PPS is somewhat appealing to the miners because the miners do not suffer mining reward variation risk, assuming that the total mining power in the system does not change.

8.5.2.2 Minimum Payout

In a pool mining activity, a miner automatically receives a payout whenever his or her balance reaches a minimum payout. A small minimum payout means that the miner will receive more frequent payouts than a higher minimum payout.

Table 8.5 shows that half of the mining pools offer a minimum payout of 0.1XMR to miners. *Minexmr* as the most significant mining pool offers a minimum payout of 0.5XMR, five times higher than the other five mining pools. On the other hand, *Moneroocean.stream* that offers the smallest minimum payout of 0.003XMR can only attract 891 miners and with a total hashrate of 1.6% of Monero network hashrate, which means that minimum payout is not essential to the miners.

We define `X-Factor` as any observable features on a mining pool's website that may increase a miner's preference to join the mining pool. Based on our observations[13], *Supportxmr* provides visual information about how many blocks the mining pool produces in a PPLNS window and how much efforts required to win each block. F2pool.com provides

[13]The observation was conducted on 19 December 2019.

Table 8.5: Monero mining pools rank list based on the pools' hashrate compared to Monero network total hashrate. Data was taken from Miningpoolstats.stream on 18 December 2019.

No	Mining Pool	Reward System	Pool Fee (%)	Min. Pay (XMR)	Num. of Miners	X-Factor	Pool Hashrate (%)
1	Minexmr.com	PPLNS	1	0.5	9,315	Simple mining calculator	37.1
2	Supportxmr.com	PPLNS	0.6	0.1	4,697	PPLNS window visualisation	28.6
3	F2pool.com	PPS	3	0.1	N/A	Android/iOS monitoring app; multicoin	11
4	Xmr.nanopool.org	PPLNS	1	0.1	1,931	Multicoin	8
5	Xmrpool.eu	PPLNS	0.9	0.1	365	N/A	4.5
6	2miners.com	SOLO	1.5	0.1	26	Multicoin	1.8
7	monerohash.com	PROP	1.6	0.5	163	Simple mining calculator	1.7
8	Moneroocean.stream	PPLNS	0	0.003	891	Auto switch to different coins.	1.6
9	Hashvault.pro	PPLNS/SOLO	0.9	0.11	675	Multicoin	1.6
10	Miningpoolhub.com	PPLNS	0.9	0.05	3,059	Auto switch to different coins.	1.1

Android and iOS applications to miners who prefer to monitor their mining performances. Interestingly, *Minexmr* that controls the biggest portion of Monero mining power (37.1% at the time of writing) does not have any noticeable `X-Factor` on its website other than a simple mining calculator. *Minexmr* uses an open-source mining pool application provided by Matthew Little[14].

8.5.2.3 Mining Pool Age and Google Trends

Table 8.6 shows the establishment date of mining pools according to their Whois registration and Wayback Machine records. *Minexmr* started to provide their service in June 2014, several months after the launch of Monero in April 2014. *Monerohash* that launched Monero mining service in February 2015 is currently only in the seventh position. *Supportxmr*, *Xmr.nanopool*, and *Xmrpool.eu* that established their Monero mining service in late 2016 are in the top five of the list. It is also notable that *F2pool* has been involved in the cryptocurrency mining industry before 2017 by offering Bitcoin mining service. At the time of writing, *F2pool* controls 16% of total Bitcoin mining power[15]. The insights from our data indicate that there is a slight correlation between mining pool age and Monero miners' preference in selecting mining pool.

We also utilised Google Trends[16] to evaluate the popularity of each mining pools. The result of a five-year Google keyword search is in Figure 8.5. *Nanopool* and *Miningpoolhub* show the best results among others. Both mining pools are multi-coin mining pools. Based on the figure, Monero miners' preference does not correlate to the popularity of the mining pools.

[14]https://github.com/zone117x/node-cryptonote-pool
[15]Data was taken from https://www.blockchain.com/pools on 19 December 2019
[16]https://trends.google.com/

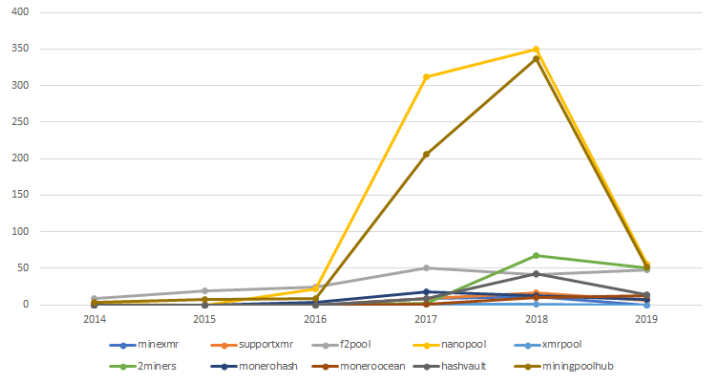

Figure 8.5: The number of Google search of mining pools in the last five years. The mining pools' domain names were used as keywords. Data was taken from Google Trends on 19 December 2019.

Table 8.6: Monero mining pools starting date according to Whois Registration and Wayback Machine. Dates marked with asterisk indicate the dates the mining pools started to offer Monero mining pool service. *Xmrpool.eu* does not have a Whois record due to EU General Data Protection Regulation [38].

No	Mining Pool	Whois Registration	Wayback Machine
1	Minexmr.com	June 2014	June 2014
2	Supportxmr.com	October 2016	December 2016
3	F2pool.com	April 2013	December 2017*
4	Xmr.nanopool.org	August 2015	August 2016
5	Xmrpool.eu	N/A	October 2016
6	2miners.com	September 2017	December 2018*
7	monerohash.com	October 2014	February 2015
8	Moneroocean.stream	August 2017	January 2018
9	Hashvault.pro	October 2017	October 2017
10	Miningpoolhub.com	January 2014	May 2017*

8.6 Conclusion and Future Work

In this chapter, we collect and analyse information published by ten Monero mining pools. The published information, including the list of won blocks and payout transactions to miners, can be utilised to trace the real outputs of the payout transactions. Our traceability analysis shows that 59.2% of inputs are traceable. We also analyse that the output age of known traceable inputs is between 2.5 hours to 3.3 days, regardless of whether the spent coins are from previous transactions or mining rewards (coinbase transactions). For the identified problems, we propose simple mitigation strategies to improve the anonymity of Monero transactions that do not require fundamental changes to Monero protocol.

We discovered little to no correlation between publishing mining-related information and miners' preference for mining pools. Most mining pools publish blocks they won and payouts to the miners, and therefore they are mostly available on the mining pools' websites. *F2pool* that does not publish the same data is currently the third-largest Monero mining pool, which indicates that Monero miners do not consider data transparency when selecting a mining pool to join. Therefore, Monero mining pools should remove the list of won blocks and payouts from their websites, because the lists leak transaction anonymity and do not provide strong business transparency to the mining pools.

We also investigated different characteristics of the top ten Monero mining pools. Our investigation includes financial factors such as reward system, mining pool fee, and minimum payout and external factors such as distinguishing features from other competitors, how long the mining pools have provided their services, and popularity through Google Trends. Each mining pool has its appeal to the miners. Miners that specialise in mining Monero would prefer *Minexmr* as one of the oldest and most established Monero mining pool in the market. *Supportxmr* that provides lower pool fee becomes the second-best Monero mining pool. The mining power distribution between the major mining pools in Monero is understandable since the community would avoid any majority mining pools and prefer a balanced mining power distribution [4].

For future work, we focus on finding a better solution to improve the transparency of their mining activities without leaking the privacy of their transactions. The solution may require modifications on Monero mining protocol.

8.7 Chapter Summary

In this chapter, we discuss how mining pools' data can reveal Monero transactions, especially related to mining payouts. We utilise two information in our traceability analyses, namely

won blocks and transaction payouts, and discover more than 200K traceable inputs. Our findings show the significant impact of mining pool's data publication to Monero transaction anonymity.

This chapter concludes our last part of the third research question, `Q3`, that focuses on investigation potential anonymity threats to third-party services, namely wallet service providers and mining pools in Monero.

Chapter 9

Conclusion and Future Work

This chapter concludes the thesis and recommends several directions for future work.

9.1 Conclusion

In this project, we work on Monero, a CryptoNote-based privacy-preserving cryptocurrency, to gain insights on privacy issues in blockchain technology, specifically at the protocol level. Cryptocurrency is one of the most successful blockchain implementations in financial technology, while Monero is a fascinating subject to research due to its rapid privacy technology adoption.

Our research contributes to the blockchain's body of knowledge by exposing vulnerabilities and potential threats and asserting the need for more protection mechanisms on the protocol level. Our findings also indicate the need of anonymity-focused improvements in the current Monero implementation that may also become a reference model for future blockchain products. To achieve our goals, we investigate three vulnerable areas in Monero that can lead to anonymity attacks.

First, our research explores how an attacker exploits Monero's scheme that relies on honest users to construct a private transaction that supposedly protects the anonymity of the sender and the receiver. We discover that an attacker could create malicious transactions that can lower other users' transaction anonymity. Our main contributions are two transaction-based attack schemes that fully comply with Monero transaction mechanisms and do not require any controls over the victims' systems [116, 117]. We also propose a traceability analysis that utilises Monero's Unencrypted Payment ID (UPID),

that results in the most significant anonymity problem after RingCT implementation [117]. These contributions answer our first research question, Q1, about vulnerabilities in Monero transaction mechanisms.

Second, the project explores the risks of Monero hard forks as the result of Monero protocol updates. The research direction is in line with our second research question, Q2. We focus on two hard forks that occur in 2018, resulting in the emerging of two new branches of Monero blockchain by the end of 2018. We discover that the hard forks are threats to Monero transaction's anonymity [119]. A hard fork in Monero also enables an attacker to launch a cheap Denial-of-Service attack to nodes that do not update their software to the latest version [118]. Our contributions in the second part of the project answer our second research question, Q2.

Third, the project studies the use of third-party services to answer our third research question, Q3. We find that a wallet service provider can trace its users' change coin by cross-referencing spent keys to previously known keys. We also discover that mining pools that play significant roles in Monero consensus model could leak their miners' transaction anonymity unnecessarily, to provide business transparency to their clients (miners). We answer our third research question by identifying potential anonymity problems from wallet service providers and mining pools.

In general, our work benefits blockchain research by providing a rigorous study on the impact of privacy-focused cryptographic protocol implementations. We identify how users' behaviors, third-party services, client-side applications, and the lack of privacy and anonymity guarantees in system and protocol levels may significantly reduce the goals of the privacy-preserving blockchain's fundamental design.

9.2 Future Work

We provide a new research direction on each part of this project for future work. In the first part, future work can provide better protection from malicious outputs by implementing a stricter output selection criteria. We suggest a new research direction for an improvement to the existing blacklisting mechanism by considering statistical properties of Monero transactions.

We also suggest exploration of alternative solutions to the current Monero hard fork mechanism to protect transaction anonymity and nodes during system migration. Regular hard forks such as the ones in Monero increase difficulties in node management. A service provider that maintains a node must halt the service, reinstall the new node, and wait for

the node, if any, to make necessary updates to the blockchain structure. The work becomes more complex if the service provider also connects the node to other applications, such as blockchain explorer or wallet application. It is likely that a major hard fork also brings major changes to the system, which, in turn, invalidates related applications. Hard forks also complicate advanced protocols such as cross-chain transactions, where a blockchain relies on oracles to provide external information to the system, such as other blockchains' block information. A hard fork in the external blockchain, especially if the hard fork produces multiple live branches, will require special strategies from the source blockchain to identify the main branch. Soft fork and velvet fork [126] are two exciting options.

In the third part of our project, we discovered how third-party services could be a threat to Monero anonymity. We suggest the development of a new thin wallet that does not rely on any third-party services to access Monero network. We also suggest working on an alternative solution to reduce or eliminate the use of mining pools in Monero.

The work we conducted in this thesis can also become a strong foundation for further research on anonymity in blockchain technology applications. We brought forward fresh approaches on identifying anonymity issues in system and protocol levels that are currently not broadly discussed. The existing solutions rely on cryptographic primitives to provide users anonymity. This approach, based on our findings, is not sufficient in live systems with more complex challenges to protect against.

References

[1] Paulo Sérgio Almeida, Carlos Baquero, Nuno Preguiça, and David Hutchison. Scalable bloom filters. *Information Processing Letters*, 101(6):255–261, 2007.

[2] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013.

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

[4] asymptotically508. Psa: Stop mining on minexmr and supportxmr. *Monero Reddit*, 2019.

[5] Khaled Baqer, Danny Yuxing Huang, Damon McCoy, and Nicholas Weaver. Stressing out: Bitcoin "stress testing". In *International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2016.

[6] Adam Barone. How to choose a cryptocurrency mining pool. *Investopedia*, 2019.

[7] BatmanLovesCrypto. Monero classic and monero original on the same blockchain? help me understand, May 2018.

[8] Morten L Bech and Rodney Garratt. Central bank cryptocurrencies. *BIS Quarterly Review September*, 2017.

[9] Belcher. Coinjoin. https://en.bitcoin.it/wiki/CoinJoin, 2015. Accessed 2018-05-07.

[10] Daniel J Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of cryptographic engineering*, 2(2):77–89, 2012.

[11] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014.

[12] Alex Biryukov and Ivan Pustogarov. Bitcoin over tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy*, pages 122–134. IEEE, 2015.

[13] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for tor hidden services: Detection, measurement, deanonymization. In *2013 IEEE Symposium on Security and Privacy*, pages 80–94. IEEE, 2013.

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

[15] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A Kroll, and Edward W Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, pages 104–121. IEEE, 2015.

[16] Boolberry. Is there information available about the Levin protocol? https://boolberry.com/design.html, 2014. Accessed: 2019-06-13.

[17] Danny Bradbury. The problem with bitcoin. *Computer Fraud & Security*, 2013(11):5–8, 2013.

[18] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 315–334. IEEE, 2018.

[19] Chris Burniske and Adam White. Bitcoin: Ringing the bell for a new asset class. *Ark Invest (January 2017) https://research. ark-invest. com/hubfs/1_Download_Files_ARK-Invest/White_Papers/Bitcoin-Ringing-The-Bell-For-A-New-Asset-Class. pdf*, 2017.

[20] Vitalik Buterin. Simple replay attack protection, Oct 2016.

[21] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[22] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, 1985.

[23] David Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of cryptology*, 1(1):65–75, 1988.

[24] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *NDSS*, 2008.

[25] Monero Classic. Upgrade announcement of xmc. http://monero-classic.org/open/notice_en.html, 2018.

[26] cmaves. 0-conf possible attack using large transactions. https://github.com/amiuhle/kasisto/issues/33, 2018.

[27] cmaves. Possible mempool spam attack. https://github.com/monero-project/monero/issues/3189, 2018.

[28] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Annual International Cryptology Conference*, pages 174–187. Springer, 1994.

[29] dEBRUYNE and Fluffypony. Long payment id deprecation. https://web.getmonero.org/2019/06/04/Long-Payment-ID-Deprecation.html, 2019. Accessed 2020-03-18.

[30] dEBRYUNE. Pow change and key reuse, Feb 2018.

[31] dEBRYUNE. Pow change and key reuse. https://ww.getmonero.org/2018/02/11/PoW-change-and-key-reuse.html, 2018.

[32] Whitfield Diffie and Martin E. Hellman. New directions in cryptography, 1976.

[33] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[34] Evan Duffield and Daniel Diaz. Dash: A privacycentric cryptocurrency. *URL: https://github.com/dashpay/dash/wiki/Whitepaper*, 2014.

[35] Justin Ehrenhofer. Monero blackball site, 2018.

[36] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and Jeremy Clark. A first look at browser-based cryptojacking. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 58–66. IEEE, 2018.

[37] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, June 2018.

[38] Anthony J Ferrante. The impact of gdpr on whois: Implications for businesses facing cybercrime. *Cyber Security: A Peer-Reviewed Journal*, 2(2):143–148, 2018.

[39] Michael Fleder, Michael S Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.

[40] Eiichiro Fujisaki and Koutarou Suzuki. Traceable ring signature. In *International Workshop on Public Key Cryptography*, pages 181–200. Springer, 2007.

[41] Florian Glaser, Kai Zimmermann, Martin Haferkorn, Moritz Christian Weber, and Michael Siering. Bitcoin-asset or currency? revealing users' hidden intentions. *Revealing Users' Hidden Intentions (April 15, 2014). ECIS*, 2014.

[42] A Greenberg. Darkwallet aims to be the anarchist's bitcoin app of choice. *URL: http://www.forbes.com/sites/andygreenberg/2013/10/31/darkwallet-aims-to-be-the-anarchists-bitcoin-app-of-choice/*, 2013.

[43] John Grunzweig. Confidential transactions. *URL: https://unit42.paloaltonetworks.com/unit42-rise-cryptocurrency-miners/ (Accessed 04/11/2019)*, 2018.

[44] Mike Hearn. Merge avoidance: A note on privacy-enhancing techniques in the Bitcoin protocol. https://medium.com/@octskyward/merge-avoidance-7f95a386692f, 2013. Accessed: 2019-06-10.

[45] Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. Tumblebit: An untrusted bitcoin-compatible anonymous payment hub. In *Network and Distributed System Security Symposium*, 2017.

[46] Ryan Henry, Amir Herzberg, and Aniket Kate. Blockchain access privacy: challenges and directions. *IEEE Security & Privacy*, 16(4):38–45, 2018.

[47] Abraham Hinteregger and Bernhard Haslhofer. An empirical analysis of monero cross-chain traceability. *arXiv preprint arXiv:1812.02808*, 2018.

[48] Abraham Hinteregger and Bernhard Haslhofer. Short paper: An empirical analysis of monero cross-chain traceability. In *International Conference on Financial Cryptography and Data Security*, pages 150–157. Springer, 2019.

[49] HitBTC. The monero original fork has happened, Apr 2018.

[50] HitBTC. Statement on monerov fork, May 2018.

[51] HitBTC. Statement on xmo monero fork, Apr 2018.

[52] Daira Hopwood, Sean Bowe, Taylor Hornby, and Nathan Wilcox. Zcash protocol specification. *Tech. rep. 2016–1.10. Zerocoin Electric Coin Company, Tech. Rep.*, 2016.

[53] jollymort. What is churning? *Monero Stackexchange*, 2017.

[54] jtgrassie. Why are there transactions in the mempool that are invalid or over 50 hours old? https://monero.stackexchange.com/a/8513, 2018.

[55] Antonio M. Juarez, Oscar Norton, Neocortex, and Albert Werner. Cryptonote standard 007: Cryptonote keys and addresses. *CryptoNote Standard*, 2012.

[56] George Kappos, Haaroon Yousaf, Mary Maller, and Sarah Meiklejohn. An empirical analysis of anonymity in zcash. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 463–477, 2018.

[57] Aggelos Kiayias, Andrew Miller, and Dionysis Zindros. Non-interactive proofs of proof-of-work. Technical report, Cryptology ePrint Archive, Report 2017/963, 2017. Accessed: 2017-10-03, 2017.

[58] knaccc. Description of a potential privacy leak and recommendation to mitigate, Feb 2017.

[59] Amrit Kumar, Clément Fischer, Shruti Tople, and Prateek Saxena. A traceability analysis of monero's blockchain. In *European Symposium on Research in Computer Security*, pages 153–173. Springer, 2017.

[60] Peter Kwang. The best monero wallets for private transactions. https://www.keysheet.io/guides/best-monero-wallets/, 2019. Accessed: 2019-07-18.

[61] David Lazar, Haogang Chen, Xi Wang, and Nickolai Zeldovich. Why does cryptographic software fail? a case study and open problems. In *Proceedings of 5th Asia-Pacific Workshop on Systems*, pages 1–7, 2014.

[62] Kevin Lee and Andrew Miller. Authenticated data structures for privacy-preserving Monero light clients. In *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 20–28. IEEE, 2018.

[63] Joseph K. Liu, Man Ho Au, Willy Susilo, and Jianying Zhou. Linkable ring signature with unconditional anonymity. *IEEE Trans. Knowl. Data Eng.*, 26(1):157–165, 2014.

[64] Joseph K Liu, Victor K Wei, and Duncan S Wong. Linkable spontaneous anonymous group signature for ad hoc groups. In *Australasian Conference on Information Security and Privacy*, pages 325–335. Springer, 2004.

[65] Adam Mackenzie, Surae Noether, and Monero Core Team. Improving obfuscation in the cryptonote protocol. *Monero Research Lab, Tech. Rep*, 2015.

[66] P Martin and M Taaki. Anonymous bitcoin transactions. *URL: https://sx.dyne.org/anontx/*, 2013.

[67] G Maxwell. Coinjoin: bitcoin privacy for the real world. *URL: https://bitcointalk.org/index.php?topic=279249.0*, 2013.

[68] G Maxwell. Coinswap: Transaction graph disjoint trustless trading. *URL: https://bitcointalk.org/index.php?topic=321228.0*, 2013.

[69] G Maxwell. I taint rich. *URL: https://bitcointalk.org/index.php?topic=139581.0*, 2013.

[70] Greg Maxwell. Confidential transactions. *URL: https://people. xiph. org/˜ greg/confidential_values. txt (Accessed 09/05/2016)*, 2015.

[71] Patrick McCorry, Ethan Heilman, and Andrew Miller. Atomically trading with roger: Gambling on the success of a hardfork. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 334–353. Springer, 2017.

[72] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kirill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: Characterizing payments among men with no names. *USENIX ;login:*, 2013.

[73] Andrew Miller, Malte Möser, Kevin Lee, and Arvind Narayanan. An empirical analysis of linkability in the monero blockchain.(2017), 2017.

[74] Monero. Monero xmr forks & hard forks. https://monero.org/forks/. [Online; accessed 4 February 2019].

[75] Monero. Monero project github page. https://github.com/monero-project/monero, 2018. [Online; accessed 4 February 2019].

[76] monero hax123. Corrupt rpc responses from remote daemon nodes can lead to transaction tracing, Mar 2018.

[77] Monero Stackexchange. Is there information available about the Levin protocol? https://monero.stackexchange.com/a/2704/9745, 2016. Accessed: 2019-06-13.

[78] Malte Möser, Rainer Böhme, and Dominic Breuker. An inquiry into money laundering tools in the bitcoin ecosystem. In *2013 APWG eCrime Researchers Summit*, pages 1–14. Ieee, 2013.

[79] Malte Möser, Kyle Soska, Ethan Heilman, Kevin Lee, Henry Heffan, Shashvat Srivastava, Kyle Hogan, Jason Hennessey, Andrew Miller, Arvind Narayanan, et al. An empirical analysis of traceability in the monero blockchain. *Proceedings on Privacy Enhancing Technologies*, 2018(3):143–163, 2018.

[80] P Carl Mullan. Liberty reserve. In *A History of Digital Currency in the United States*, pages 171–196. Springer, 2016.

[81] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin.org*, 2008.

[82] Sarang Noether and Brandon Goodell. An efficient implementation of monero subaddresses, mrl-0006, october 2017, 2017.

[83] Shen Noether, Adam Mackenzie, et al. Ring confidential transactions. *Ledger*, 1:1–18, 2016.

[84] Shen Noether and Sarang Noether. Monero is not that mysterious. *Technical report*, 2014.

[85] Surae Noether, Sarang Noether, and Adam Mackenzie. Mrl-0001: A note on chain reactions in traceability in cryptonote 2.0. *Technical report2014*, 2014.

[86] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013.

[87] Sergio Pastrana and Guillermo Suarez-Tangil. A first look at the crypto-mining malware ecosystem: A decade of unrestricted wealth. *arXiv preprint arXiv:1901.00846*, 2019.

[88] Andreas Pfitzmann and Marit Hansen. A terminology for talking about privacy by data minimization: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management. *maroki.de*, 2010.

[89] Piuk. What is taint? *URL: https://bitcointalk.org/index.php?topic=92416.msg1018943#msg1018943*, 2012.

[90] propercoil. Replay protection? *URL: https://www.reddit.com/r/Monero/comments/8agjfd/replay_protection/dx0lun4/*, 2018.

[91] Jeffrey Quesnelle. On the linkability of zcash transactions. *arXiv preprint arXiv:1712.01210*, 2017.

[92] Fergal Reid and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer, 2013.

[93] Bailey Reutzel. Logical or not, bitcoin's coming fork is boosting its price, Oct 2017.

[94] Ronald L Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565. Springer, 2001.

[95] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security*, pages 6–24. Springer, 2013.

[96] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474. IEEE, 2014.

[97] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

[98] sgp. How can individuals safeguard themselves and the community against a key reusing fork?, Mar 2018.

[99] Cecilia Skingsley. Should the riksbank issue e-krona? *speech at FinTech Stockholm*, 16, 2016.

[100] Riccardo Spagni. Monero 0.13.0 "beryllium bullet" release, Oct 2018.

[101] Shifeng Sun, Man Ho Au, Joseph K. Liu, and Tsz Hon Yuen. Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In *ESORICS II*, volume 10493 of *LNCS*, pages 456–474. Springer, 2017.

[102] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[103] Peter Todd. Stealth addresses. *URL: http://sourceforge.net/p/bitcoin/mailman/message/31813471/*, 2014.

[104] unSYSTEM Wiki. Dark wallet/stealth. *URL: https://wiki.unsystem.net/en/index.php/DarkWallet/Stealth*, 2014.

[105] user36303. Replay attack and cryptonotes, Aug 2017.

[106] user36303. How can individuals safeguard themselves and the community against a key reusing fork?, Mar 2018.

[107] Marshall Van Alstyne. Why bitcoin has value. *Communications of the ACM*, 57(5):30–32, 2014.

[108] Nicolas van Saberhagen. Cryptonote v 2.0, 2013. *CryptoNote*, pages 04–13, 2013.

[109] Nicolas van Saberhagen, Johannes Meier, Antonio M. Juarez, Max Jameson, and Seigen. Cryptonote standard 003: Cryptonote blokchain. *CryptoNote Standard*, 2012.

[110] Nicolas van Saberhagen, Seigen, Johannes Meier, and Richard Lem. Cryptonote standard 006: Cryptonote one-time keys. *CryptoNote Standard*, 2012.

[111] Marie Vasek, Micah Thornton, and Tyler Moore. Empirical analysis of denial-of-service attacks in the bitcoin ecosystem. In *International conference on financial cryptography and data security*, pages 57–71. Springer, 2014.

[112] MGCSA Walport et al. Distributed ledger technology: Beyond blockchain. *UK Government Office for Science*, 1, 2016.

[113] Albert Weiner, Montag, Ardolabar, Tereno, and Antonio M. Juarez. Cryptonote standard 003: Cryptonote blokchain. *CryptoNote Standard*, 2012.

[114] Albert Weiner, Montag, Prometheus, and Tereno. Cryptonote standard 005: Cryptonote transaction extra field. *CryptoNote Standard*, 2012.

[115] Albert Weiner, Kenji Sugihara, Montag, Ardolabar, Tereno, and Antonio M. Juarez. Cryptonote standard 004: Cryptonote transactions. *CryptoNote Standard*, 2012.

[116] Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, and Dongxi Liu. Monero ring attack: Recreating zero mixin transaction effect. In *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pages 1196–1201. IEEE, 2018.

[117] Dimaz Ankaa Wijaya, Joseph Liu, Ron Steinfeld, Dongxi Liu, and Tsz Hon Yuen. Anonymity reduction attacks to monero. In *The 14th International Conference on Information Security and Cryptology*. Springer, 2018.

[118] Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, and Dongxi Liu. Risk of asynchronous protocol update: Attacks to monero protocols. In Julian Jang-Jaccard and Fuchun Guo, editors, *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, volume 11547 of *Lecture Notes in Computer Science*, pages 307–321. Springer, 2019.

[119] Dimaz Ankaa Wijaya, Joseph K. Liu, Ron Steinfeld, Dongxi Liu, and Jiangshan Yu. On the unforkability of monero. In Steven D. Galbraith, Giovanni Russello, Willy Susilo, Dieter Gollmann, Engin Kirda, and Zhenkai Liang, editors, *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security, AsiaCCS 2019, Auckland, New Zealand, July 09-12, 2019*, pages 621–632. ACM, 2019.

[120] Dimaz Ankaa Wijaya, Joseph K Liu, Ron Steinfeld, Shi-Feng Sun, and Xinyi Huang. Anonymizing bitcoin transaction. In *International Conference on Information Security Practice and Experience*, pages 271–283. Springer, 2016.

[121] Dimaz Ankaa Wijaya, Joseph K Liu, Dony Ariadi Suwarsono, and Peng Zhang. A new blockchain-based value-added tax system. In *International Conference on Provable Security*, pages 471–486. Springer, 2017.

[122] Dimaz Ankaa Wijaya and Dony Ariadi Suwarsono. Securing digital evidence information in bitcoin. Technical report, Technical report, Monash University Melbourne, Australia, 2016.

[123] Pieter Wuille. Hierarchical deterministic wallets. *Online specification for BIP32 at https://github.com/bitcoin/bips/blob/master/bip-0032. mediawiki*, 1:K2, 2012.

[124] Bin Yu, Joseph K. Liu, Amin Sakzad, Surya Nepal, Ron Steinfeld, Paul Rimba, and Man Ho Au. Platform-independent secure blockchain-based voting system. In *ISC*, volume 11060 of *LNCS*, pages 369–386. Springer, 2018.

[125] Zuoxia Yu, Man Ho Au, Jiangshan Yu, Rupeng Yang, Qiuliang Xu, and Wang Fat Lau. New empirical traceability analysis of cryptonote-style blockchains. In *Financial Cryptography and Data Security*, 2019.

[126] Alexei Zamyatin, Nicholas Stifter, Aljosha Judmayer, Philipp Schindler, Edgar Weippl, and WJ Knottebelt. A wild velvet fork appears! inclusive blockchain protocol changes in practice. In *5th Workshop on Bitcoin and Blockchain Research, Financial Cryptography and Data Security*, volume 18, 2018.