

BitIodine: Extracting Intelligence from the Bitcoin Network

BY

Michele Spagnuolo

Laurea, Politecnico di Milano, Milan, Italy, July 2011

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2013

Chicago, Illinois

Defense Committee:

V.N. Venkatakrisnan, Chair and Advisor

Bhaskar DasGupta

Stefano Zanero, Politecnico di Milano

*To anyone who ever taught me,
and to anyone I ever learned from.*

ACKNOWLEDGEMENTS

My first debt of gratitude is due to my PoliMi advisor Stefano Zanero for his help, guidance and general kindness. I owe him my heartfelt appreciation. Discussions with Stefano and Federico Maggi greatly contributed to the exposition and development of this thesis.

Thanks to V.N. Venkatakrishnan, UIC advisor, who dedicated his time in giving me an overview of the computer security academic world.

I would like to thank the following friends, fellow students and teachers who have been source of thoughts, inspiration and smiles. In random order (the list was scrambled using `random.org` List Randomizer¹, with randomness coming from atmospheric noise): Francesca Prosperuzzi, Edoardo Colombo, Elena Porta, Alessandra Piatti, Francesca Elisa Diletta Raimondi, Francesca Ragni, Matteo Paglino, Elisa Dui, Mario Sangiorgio, Diego Martinoia, Alessandro Frossi, Stefano Schiavoni, Davide Tessaro, Gabriele Petronella, Matteo Serva, Marco D. Santambrogio, Daniele Galligani, Cristina Palamini, Alessandro Barengni, Matteo Turri, Luciano Righi, Giuseppina Ferolo, Alberto Scolari, Giulia Cesana, Giovanni Gonzaga, Luca Cioria.

Special thanks to Giorgio Cavaggion, with whom I shared the Chicago experience, for his amazing awesomeness.

¹<http://www.random.org/lists/>

ACKNOWLEDGEMENTS (Continued)

Thanks to the awesome people at NECSTLab in Milan, and to the Bitcoin, Gephi and Python communities.

A *thank you* to the mysterious Satoshi Nakamoto, the mastermind behind Bitcoin.

I wish to thank my parents, Laura and Pasquale, my inspiration and driving force.

My love Liz, who simply is the best thing that ever happened to me.

Finally, I would like to dedicate this work to my grandma Carla.

MS

TABLE OF CONTENTS

<u>CHAPTER</u>	<u>PAGE</u>
1 INTRODUCTION	1
2 STATE OF THE ART	3
2.1 An overview of Bitcoin	3
2.1.1 History of Bitcoin	3
2.1.2 What is Bitcoin?	4
2.1.3 How Bitcoin works	5
2.1.4 Exchanges	8
2.1.5 Regulation in the United States	9
2.1.6 Reception and usage	10
2.1.7 Trading goods for Bitcoin	11
2.1.8 The Silk Road	12
2.1.9 Interest in the financial world	12
2.1.10 Bitcoin in economic theories	13
2.1.11 Cyprus effect	14
2.1.12 Bitcoin ATMs	14
2.1.13 Bitcoin related malware	15
2.1.14 Privacy in Bitcoin	15
2.2 Previous related works	17
3 DESIGN	18
3.1 Bitcoin terminology	18
3.1.1 Bitcoin address	18
3.1.2 Wallet	18
3.1.3 Blockchain	18
3.1.4 Block	19
3.1.5 Transaction	19
3.2 Building blocks of BitIodine	20
3.2.1 The block parser	21
3.2.2 The clusterizer	21
3.2.3 The classifier	22
3.2.4 The scrapers	22
3.2.5 The Mt. Gox scraper	23
3.2.6 The grapher	23
3.2.7 The exporters	23
3.3 Transaction and User graphs	24

TABLE OF CONTENTS (Continued)

<u>CHAPTER</u>	<u>PAGE</u>
4 IMPLEMENTATION	28
4.1 Code structure	29
4.2 Building the User Graph	30
4.2.1 A formal model	30
4.2.2 The first heuristic: Multi-input transactions grouping	31
4.2.3 The second heuristic: shadow address guessing	32
4.2.3.1 Variant 1 – deterministic version	33
4.2.3.2 Variant 2 – exploiting a logical flaw in the official bitcoin client	34
4.2.4 Evaluating heuristics	36
4.2.4.1 First heuristic	37
4.2.4.2 Second heuristic	37
4.3 Employed technologies	41
4.3.1 SQLite databases	41
4.3.1.1 Blockchain DB schema	41
4.3.1.2 Features DB schema	42
4.3.1.3 Trades DB schema	43
4.3.2 NetworkX graphs	44
4.4 Using the Classifier	44
4.4.1 Exporting and visualizing	46
5 EXPERIMENTAL RESULTS	48
5.1 Use cases: classifying addresses and users	49
5.1.1 A real-world case: investigating the Silk Road	53
5.1.2 Another real-world case: the Slush mining pool hack	60
5.1.3 An expensive pizza	61
6 CONCLUSIONS	64
APPENDICES	65
Appendix A	66
Appendix B	70
Appendix C	72
Appendix D	74
CITED LITERATURE	77
VITA	79

LIST OF TABLES

<u>TABLE</u>		<u>PAGE</u>
I	TRANSACTION COVERAGE OF DIFFERENT HEURISTICS	37
II	CLUSTERING WITH FIRST HEURISTIC ONLY	38
III	CLUSTERING WITH HEURISTICS 1 AND 2 (DETERMINISTIC) .	39
IV	CLUSTERING WITH HEURISTICS 1 AND 2	40
V	USE CASE: ADDRESS LABELS	52
VI	USE CASE: CLUSTER LABELS	52
VII	LABELS FOR ADDRESSES	67
VIII	LABELS FOR CLUSTERS	69

LIST OF FIGURES

<u>FIGURE</u>		<u>PAGE</u>
1	A tree representation of the Bitcoin blockchain	6
2	Building blocks of BitIodine	20
3	A transaction graph – transactions related to a Bitcoin faucet . . .	25
4	A graph with addresses grouped in users – SatoshiDice gamblers .	27
5	Code structure of BitIodine	29
6	Statistics about clusters while evaluating heuristics	36
7	Transaction count per number of outputs	38
8	Transaction volume per number of outputs	39
9	SQL schema diagram for the blockchain representation	42
10	SQL schema diagram for the features DB	43
11	SQL schema diagram for the trades DB	44
12	Hybrid graph of transactions of more than 100 BTC	47
13	Classifying an address with BitIodine	50
14	GPG signed contract of a defaulted loan, and IRC chat logs	51
15	Plot of balance over time of a Silk Road-owned address	55
16	Bitcoin price chart for 17 Jul 2012	56
17	Two big trades on the Mt. Gox exchange	57
18	An important transaction in our Silk Road investigation	58
19	The transaction that links the address to Silk Road	59
20	Graph of transactions inside the <i>laszlo</i> cluster	62
21	Screenshot posted by <i>laszlo</i> of his wallet	63

LIST OF ABBREVIATIONS

ASIC Application-specific integrated circuit – an integrated circuit customized for a particular use, rather than intended for general-purpose use. viii

BTC Bitcoin. viii

DB Database. viii

ECDSA Elliptic Curve Digital Signature Algorithm (DSA) – a variation of DSA based on calculations of elliptical curves over finite space. viii

FPGA Field-programmable gate array – an integrated circuit designed to be configured by a customer or a designer after manufacturing. viii

GPU Graphics processing unit – a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the building of images in a frame buffer intended for output to a display. viii

SUMMARY

Anonymity in Bitcoin, a peer-to-peer, decentralized electronic currency system, is a complicated issue. While the Bitcoin technology can support strong anonymity, the current implementation is not strongly anonymous.

In this thesis we present a tool, BitIodine, for extracting intelligence from the Bitcoin network, by grouping *transaction graphs* into *user graphs*, thanks to heuristics that set ownership relations between addresses and users, and profiling activity at different abstraction levels: the transaction level, the address level, and up to the user level.

BitIodine makes use of a new approach for creating the *user graph*, and is actually a collection of deployable modules to parse the blockchain, cluster addresses, classify addresses and users, graph, export and visualize elaborated information from the Bitcoin network.

The BitIodine architecture is meant to be modular, expandable and easily deployable. Using BitIodine, it is easy to detect gamblers, miners or known scammers. Thanks to web scrapers, users may also be recognized as specific BitcoinTalk (forum) or Bitcoin-OTC (exchange) users.

We will test our tool on several previously known and unknown use cases, and we demonstrate that we are able to get valuable information.

Thanks to its architecture, we believe BitIodine can be used in the future as a skeleton for building more complex frameworks for Bitcoin forensic analysis.

CHAPTER 1

INTRODUCTION

Bitcoin is a decentralized global digital currency that aims to become the digital equivalent of cash. Based on an open source, peer-to-peer network, Bitcoin has recently received important media coverage, mostly due to its price in dollars surging more than 12,000% in four months and the bitcoin economy expanding at unprecedented pace, while the economic crisis is dominating the global scenario.

While the Bitcoin technology can support strong anonymity, the current implementation is not strongly anonymous, as we discuss in Section 2.1.14.

In this thesis we present BitIodine, a modular, expandable and easily deployable framework for de-anonymizing transactions on the Bitcoin network, thanks to heuristics that set ownership relations between addresses and users, and for profiling activity at different abstraction levels.

We exploit a bug in existing Bitcoin client implementations to get better clustering, thus creating a more compact user graph with respect to previous works (see Section 2.2). We also implement fuzzy *labelling* (or *tagging*) of users and addresses, in order to profile transactions, addresses and, finally, users in the Bitcoin network.

For example, BitIodine can be used for finding miners, or gamblers. Users may also be recognized as specific BitcoinTalk or Bitcoin-OTC users, thanks to scrapers that automatically crawl forum signatures and databases.

In Chapter 2 we will go through an high-level overview of Bitcoin, and describe the state of the art and previous related works. Chapter 3 is devoted to architectural choices and building blocks of BitIodine, as well as some basic concepts. We get into implementation aspects in Chapter 4, where we formalize the important concepts and describe our implementation in detail. Use cases and results are in Chapter 5, followed by the conclusions in Chapter 6.

CHAPTER 2

STATE OF THE ART

2.1 An overview of Bitcoin

Bitcoin (often referred to as BTC) is a decentralized global digital currency based on an open source, peer-to-peer network that creates a time-stamped register yielding chains of valid transactions. Bitcoin aims to be a digital equivalent of cash, and its security is founded on distributed cryptography. The network is programmed to increase the money supply according to a geometric series until the total number of bitcoins reaches 21 million BTC in slightly more than 100 years.

2.1.1 History of Bitcoin

The need for a digital currency based in cryptography was discussed in two separate academic papers published in 1993 by researchers at Carnegie Mellon University⁽¹⁾ and the University of Southern California⁽²⁾. Five years later, cryptography advocate Wei Dai suggested a system in which the currency would be both regulated and created through crowdsourced cryptography, thus eliminating the risk of double-spending altogether⁽³⁾.

On November 1st, 2008, a person or group of people under the pseudonym Satoshi Nakamoto distributed a paper⁽⁴⁾ solidifying this idea into a proposal for peer-to-peer electronic cash system called *Bitcoin*. Beyond Bitcoin, no other links to this identity have been found, and his involvement in the original bitcoin protocol does not appear to extend past

mid-2010. The first blockchain (see Section 3.1.3) was generated on or after January 3rd, 2009, as its first block (see Section 3.1.4), called *genesis block*, references the title of an article published that day in the UK newspaper *The Times* about a bank bailout⁽⁵⁾. The announcement of the system and its open source client was posted on the Cryptography Mailing List¹ on January 9th.

2.1.2 What is Bitcoin?

While Bitcoin can be seen as a *trust-no-one* currency that isn't subject to manipulation by central banks or corporations, from a more technical point of view it is a payment system written from scratch and based on the very best cryptography, designed for security. The Bitcoin protocol has been designed to be forward compatible and *future proof* (more details in the following subsection).

Open source and resilient to attacks, having no single point of failure, it is very difficult to take down.

There are several advantages of using Bitcoin as a means of exchange:

Speed and price It is possible to transfer money anywhere in the world within minutes with negligible fees.

No central authority Bitcoin is not dependent on any company or government to maintain its value.

¹<http://www.mail-archive.com/cryptography@metzdowd.com/msg10142.html>

No setup Merchants can start accepting bitcoins instantaneously, without setting up merchant accounts, buying credit card processing hardware, etc.

Better privacy Bitcoins are less traceable than many types of monetary transactions (though not anonymous, as we discuss in this thesis).

No counterfeit, no chargebacks Bitcoins can not be counterfeited and transactions can not be reversed.

No account freezing No transaction blocking or account freezing. Governments can freeze bank accounts of dissidents and payment processors refuse to process certain types of transactions (for example, PayPal froze WikiLeaks' account⁽⁶⁾).

Algorithmically known inflation Bitcoin is seen as a store of value because the total number of bitcoins that will ever be created is known in advance and it is impossible to create more than that.

2.1.3 How Bitcoin works

Bitcoin makes use of several cryptographic technologies.

First is public key cryptography. Technically, bitcoin addresses are nothing more than hashed public keys. Each coin is associated with its current owner's public ECDSA (Elliptic Curve Digital Signature Algorithm, based on calculations of elliptical curves over finite space) key.

When a user sends bitcoins to someone, a message (*transaction*) is created, attaching the new owner's public key to this amount of coins, and signing it with the payer's pri-

vate key. When the transaction is broadcast to the bitcoin network, the signature on the message verifies for everyone that the transaction is authentic. The complete history of transactions is kept by everyone, so anyone can verify who is the current owner of any particular group of coins.

This complete record of transactions is kept in the *block chain*, which is a sequence of records called *blocks*. All computers in the peer-to-peer network have a copy of the block chain, which they keep updated by passing along new blocks to each other. Each block contains a group of transactions that have been sent since the previous block. In order to preserve the integrity of the block chain, each block in the chain confirms the integrity of the previous one, all the way back to the first one, the genesis block.

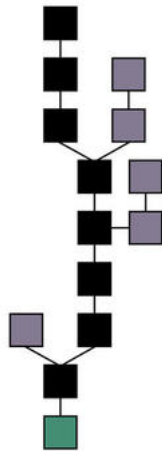


Figure 1: A tree representation of the Bitcoin blockchain

For any block on the chain, there is only one path to the genesis block. Coming from the genesis block, however, there can be forks. One-block forks are created from time to time when two blocks are created just a few seconds apart. When that happens, generating nodes build onto whichever one of the blocks they received first.

The generation of a new block is costly because each block must meet certain requirements that make it difficult to generate a valid block. To make generating bitcoins difficult a *cost-function* is used.

Integrity, block-chaining, and the cost-function rely on SHA (SHA-256 and SHA-1) as the underlying cryptographic hash function.

Blocks can be identified by their hash, which serves the dual purpose of identification as well as integrity verification. The cost-function difficulty factor is achieved by requiring that the hash output has a number of leading zeros, an easily-verifiable proof of work – every node on the network can instantly verify that a block meets the required criteria. This framework guarantees the essential features of the Bitcoin system: verifiable ownership of bitcoins and a distributed database of all transactions, which prevents *double spending*.

Users enter the Bitcoin system by trading non-digital currencies for the Bitcoin currency at a small number of currency exchange marketplaces, or by *mining* coins, where miners computationally compete to solve a cryptographically hard problem; the solver is then rewarded a fixed number of bitcoins for cryptographically validating transactions on the network.

This *mining* is somehow analogous to physical mining of gold. Rather than being backed by gold or other precious metals, the value of bitcoins is derived from computation expended.

Nowadays mining with GPUs is no longer profitable, and FPGAs or ASICs are used. Companies such as Butterfly Labs¹ manufacture high speed processors and boards dedicated to bitcoin mining.

Finally, Bitcoin protocol has been designed to be forward compatible. While ECDSA is not secure under quantum computing, quantum computers of a kind that could be used for cryptography do not exist yet. Bitcoin's security, when used properly with a new address on each transaction, depends on more than just ECDSA: cryptographic hashes such as SHA-256 are much stronger than ECDSA under a quantum computing paradigm. Bitcoin's security was designed to be upgraded in a forward compatible way and provides a scripting language layer that makes switching to other algorithms transparent to users, if this were considered an imminent threat.

2.1.4 Exchanges

On July 18th, 2010, the Japan-based exchange market *Mt. Gox* launched, allowing people to buy and sell Bitcoins in exchange for real currencies such as US dollars or Euro, as well as providing a simple way for merchants to accept Bitcoins as payment on their websites. They claim to facilitate more than 80% of all Bitcoin trading as of July 2011.

¹<http://www.butterflylabs.com>

Recently, Mt. Gox joined forces with Coinlab - the world's first US-venture backed Bitcoin company - to cater to their customer base in the US and Canada.

There are several other minor exchanges, mostly based in Europe and the US. Bitcoins can also be traded locally from person to person, using services such as LocalBitcoins¹.

At the time of writing (April 2013), Bitcoin is in high demand, one Bitcoin trades for 250 USD (more than twelve times the price in January of the same year), and its value increased sixfold in a month. The network has a market capitalization of more than 2.5 billion USD.

2.1.5 Regulation in the United States

On March 18th, 2013, the United States Financial Crimes Enforcement Network issued a clarification⁽⁷⁾ to the US regulation regarding virtual currencies.

The statement, without explicitly addressing Bitcoin, stipulates that digital currencies are to be treated essentially as foreign currencies, and not as tender money. This clarifies that Bitcoin will not be treated as illegal tender in the US, because, according to FinCEN, it lacks all the real attributes of real currency. Companies that exchange bitcoins for real money will have to comply with the same money laundering regulations as traditional currency exchangers – namely, they must verify the identity of anyone exchanging money for bitcoins and report large transactions to the government.

Using a digital currency to purchase goods, however, is specifically exempted.

¹<https://localbitcoins.com>

2.1.6 Reception and usage

Bitcoin users and miners congregate on Reddit and the Bitcoin Talk Forums, among numerous other smaller local and regional groups. There is also a Wiki¹ and the online publication Bitcoin Magazine² that gathers information about the currency. Since 2011, Bitcoin conferences have been held annually throughout Europe (2011, Prague and 2012, London), with the first US conference scheduled for May 2013 in San Jose, California.

A number of online businesses and non-profit organizations accept Bitcoins, most notably Wordpress⁽⁸⁾, 4chan⁽⁹⁾, Wikileaks⁽¹⁰⁾, Reddit⁽¹¹⁾, domain registrar and hosting provider Namecheap⁽¹²⁾ and one of the biggest online dating communities, OkCupid. Additionally, the Internet Archive has offered their employees an option to receive a portion of their paychecks in Bitcoins⁽¹³⁾.

BitElectronics³ is an e-commerce website that sells consumer electronics to all EU countries in Bitcoin, with free shipping.

Howard Johnson, a chain of hotels and restaurants located primarily throughout the United States and Canada, accepts Bitcoin.

Online food delivery and takeout portal Foodler is accepting Bitcoin alongside credit cards and cash-on-delivery for orders from more than 17,000 restaurants in the US.

¹https://en.bitcoin.it/wiki/Main_Page

²<http://bitcoinmagazine.com/>

³<http://bitelectronics.net>

There are also a handful of Bitcoin casinos and gambling sites. The transparent nature of Bitcoin, where every transaction is public, revolutionized the online gambling world, since owners can provide cryptographically provable fairness and publicly display proof of payment of winnings. Finally, a number of truly zero-sum games where players compete against each other, and not against the gambling site, recently appeared.

2.1.7 Trading goods for Bitcoin

Bitcoin, especially in recent months, is being used by individuals to trade goods.

For example, Bitmit¹ is an *eBay*-like shopping platform on which people from all over the world can trade their goods using Bitcoin. It is also simple to buy gift cards for Amazon or other big e-commerce companies.

We would like to report two interesting examples of trading between individuals: the first takes place in May 2010, the second in April 2013.

In May of 2010, a BitcoinTalk user called *laszlo* from Jacksonville, Florida, bought two pizzas for 10,000 BTC². Another user, *jercos*, bought two pizzas to be delivered to him and posted photos as proof³⁴.

¹<http://bitmit.net>

²<https://bitcointalk.org/index.php?topic=137.0>

³<http://heliacal.net/~solar/bitcoin/pizza/>

⁴<http://blockchain.info/tx/a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d>

10,000 BTC were valued \$41 at the time of the trade. In April 2013, they can be sold for 2.5 million USD.

We will analyze this trade with BitIodine in Chapter 5.

On April 2, 2013 a family in Austin sold a 2007 Porsche Cayman S for 300 BTC¹. The buyer reportedly purchased his initial Bitcoin investment years ago, at around \$4 a piece, thus actually paying the car \$1200 only (300 BTC were valued around 42,000 USD at the time of the trade).

2.1.8 The Silk Road

Bitcoins are the only currency accepted on Silk Road, an online black market in the *deep web* that can only be accessed via TOR. Even though the site launched in February 2011, the site did not receive mainstream attention until Gawker published an expose on it in June of that year. Silk Road allows people to buy a number of items including drugs, apparel, books, digital goods, drug paraphernalia, erotica and forgeries. A recent study estimates that 23 million USD of illicit items are sold for bitcoins on the site every year⁽¹⁴⁾.

2.1.9 Interest in the financial world

In 2013, Exante Ltd., a Malta-based investment firm, opened to the public a bitcoin hedge fund marketed towards institutional investors and high net-worth individuals. Bitcoin shares are currently traded through the Exante Hedge Fund Marketplace platform

¹<https://bitcointalk.org/index.php?topic=143722.0>

and authorized and regulated by the Malta Financial Services Authority. As of March 2013, Exante holds 3.2 million USD in bitcoin assets.

2.1.10 Bitcoin in economic theories

Will Bitcoin ever become mainstream money?

While this is an important question, it is also not relevant for the broader question of the future of Bitcoin.

The *money or nothing fallacy* argues that if we can refute that Bitcoin is or will ever be *money*, it follows that it is *nothing*.

We think that, as long as it provides a significant advantage in transaction costs, Bitcoin will be competitive, even if it will never replace fiat currencies, whether that is due to too much leverage the states have over fiat money, or due to inertia. Non-Austrians economists have called such a medium of exchange *metacurrency*, for example, Krugman calls it *vehicle currency*⁽¹⁵⁾. Austrians also have a name suitable for such class of media of exchange, for example *secondary media of exchange*⁽¹⁶⁾ (Mises) or *quasi-monies*⁽¹⁷⁾ (Rothbard).

From the economic point of view, Bitcoin has the following features:

- **immaterial good**
- with **ultra low transaction costs**, and
- **inelastic supply** (regardless of price or demand, a fixed amount of bitcoins are generated every ten minutes.)

Since Bitcoin is not a *claim* (nobody is obligated to redeem it), nor is it treated at par with anything else, it should not be considered a *money substitute*, but it is closer to *commodity money*. Another reason for classifying Bitcoin as commodity money is the inelasticity of supply.

2.1.11 Cyprus effect

Starting from the end of March 2013, depositors in the island nation of Cyprus have been struggling to gain access to cash stored in banks, and some face losses on their deposits.

The situation has caused investors all over Europe to question the safety of the banking system, and Cyprus began discussing tapping deposits as part of the bailout by the EU and IMF.

This probably led bitcoin price to surge, since investors were looking for diversification and a safe haven from governmental interference.

2.1.12 Bitcoin ATMs

At the time of writing, 42-year-old media entrepreneur Jeff Berwick expects to put the first two Bitcoin ATMs in Los Angeles and Cyprus in the next two weeks and is choosing between several different retail locations in both areas. He added that orders are coming in by the hundreds from 30 different countries⁽¹⁸⁾.

2.1.13 Bitcoin related malware

In July 19, 2011, the first known Bitcoin miner trojan was found in the wild by Symantec, and named *Trojan.Coinbitminer*¹. A month after, another Bitcoin mining bot, controlled via Twitter, was found by F-Secure⁽¹⁹⁾.

In April 2013, according to antivirus seller Kaspersky Lab, a new trojan that takes control of infected machines and forces them to mine Bitcoin is spreading via Skype. With the trojan, cybercriminals aim at creating a botnet of infected machines, called *zombies*, forced to perform complex calculations to earn them money, putting the machines under heavy CPU and GPU load.

2.1.14 Privacy in Bitcoin

In 2012, *Androulaki, Elli et alii*⁽²⁰⁾ explored the privacy implications of Bitcoin. Their findings show that the current measures adopted by Bitcoin are not enough to protect the privacy of users if Bitcoin were to be used as a digital currency in realistic settings. More specifically, in a small controlled environment, clustering techniques were found suitable to unveil the profiles of Bitcoin users, even if these users try to enhance their privacy by manually creating new addresses.

The main problem, when it comes to anonymity, is that the history of a coin is publicly available. Anyone can see the flow of bitcoins from address to address. Common anonymization techniques are:

¹https://www.symantec.com/security_response/writeup.jsp?docid=2011-072002-1302-99

- Randomly sending coins to new addresses generated just for this purpose. The coins are still part of the owner's balance, but it is very difficult for an outsider to prove that the owner sent the coins to himself instead of another person. However, the transaction chain still has the owner's identity in it.
- Using a *mixer*, that takes the coins of many different people, mix them up, and send similar amounts back to those peoples' addresses. If the mixer keeps no logs of who gets which coins, investigations are unfeasible.

To be truly anonymous, Bitcoin should have, by default, the capability to automatically send coins through several external mixers.

Bitcoin is also vulnerable to network analysis attacks: if an attacker is able to watch all of the victim's incoming and outgoing traffic, (s)he can easily see which transactions are initiated by the victim. Since the connection is not encrypted, transactions broadcast (and not received) by the victim are the ones originated by the victim.

Records of every single Bitcoin transaction that is ever been conducted are public. From a privacy perspective, this is a weakness. Due to the way Bitcoin works, this information can not be limited to just a few trustworthy parties, since there are *no* trustworthy parties. This means all of your transactions are conducted in public, and each transaction is tied to the one that precedes it. In a sense this makes Bitcoin much less private than *cash*, and even worse than *credit cards*. If an user chooses to engage in sensitive transactions on Bitcoin, (s)he should be aware that a public record will be preserved forever.

Since every user can generate as many addresses as (s)he wants, Bitcoin offers privacy through *pseudonymity* only, and in this thesis we show how it is possible to de-anonymize Bitcoin transactions.

Very recently, in 2013, Ian Miers, Christina Garman, Matthew Green and Aviel D. Rubin proposed a cryptographic extension to Bitcoin that augments the protocol to allow for fully anonymous transactions called Zerocoin⁽²¹⁾. To achieve the goal, Zerocoin adds extensions to the existing Bitcoin protocol, such as digital commitments (that allow one to commit to a chosen statement while keeping it hidden to others, with the ability to reveal the committed statement later), one-way accumulators (a decentralized alternative to digital signatures) and zero-knowledge proofs (a method by which one party, the *prover*, can prove to another party, the *verifier*, that a given statement is true, without conveying any additional information).

2.2 Previous related works

The idea of grouping transaction networks into user networks is described by *Androulaki, Elli et alii*⁽²⁰⁾ and *Ivan Brugere*⁽²²⁾ in their cited works. *Reid* and *Harrigan* also carried out an important analysis of anonymity in the Bitcoin system⁽²³⁾, while *Nicolas* collected precious information about the Silk Road⁽¹⁴⁾.

CHAPTER 3

DESIGN

In this chapter we first informally present the essential terminology that will be used in this thesis to describe Bitcoin and BitIodine concepts. We then show the building blocks for our tool BitIodine, and detail the purpose and functionalities of every block.

3.1 Bitcoin terminology

3.1.1 Bitcoin address

A *Bitcoin address* is a string like 1MikiSPbrhCFk7S4wzZP7gQqhWH866DCb generated by a Bitcoin client together with the private key needed to redeem the coins sent to it. It is public, and can be posted everywhere in order to receive payments.

3.1.2 Wallet

A *wallet* is a file which stores addresses and the private keys needed to use them.

3.1.3 Blockchain

The *blockchain* is a shared public transaction log on which the entire Bitcoin network relies. All confirmed transactions are included in the blockchain with no exception. This way, new transactions can be verified to be spending bitcoins that are actually owned by the spender. The integrity and the chronological order of the blockchain are enforced with cryptography¹.

¹<http://bitcoin.org/en/how-it-works>

3.1.4 Block

A *block* is an individual unit of a blockchain. In order to guarantee integrity, each block contains the hash of the previous block and as many unconfirmed (not embedded in previous blocks) transactions as can be found in the network.

3.1.5 Transaction

A *transaction* is a transfer of value between Bitcoin addresses that gets included in the *blockchain* and broadcast by the network. Transactions are *signed* by private keys of owners of the input addresses, providing a mathematical proof that they come from the owner of the addresses. The signature also prevents the transaction from being altered by anybody once it has been issued.

3.2 Building blocks of BitIodine

Our aim is to design a modular, expandable and easily deployable framework for analyzing activity in the Bitcoin network.

Here is a schema of the building blocks of BitIodine. We will detail each of them in this section.

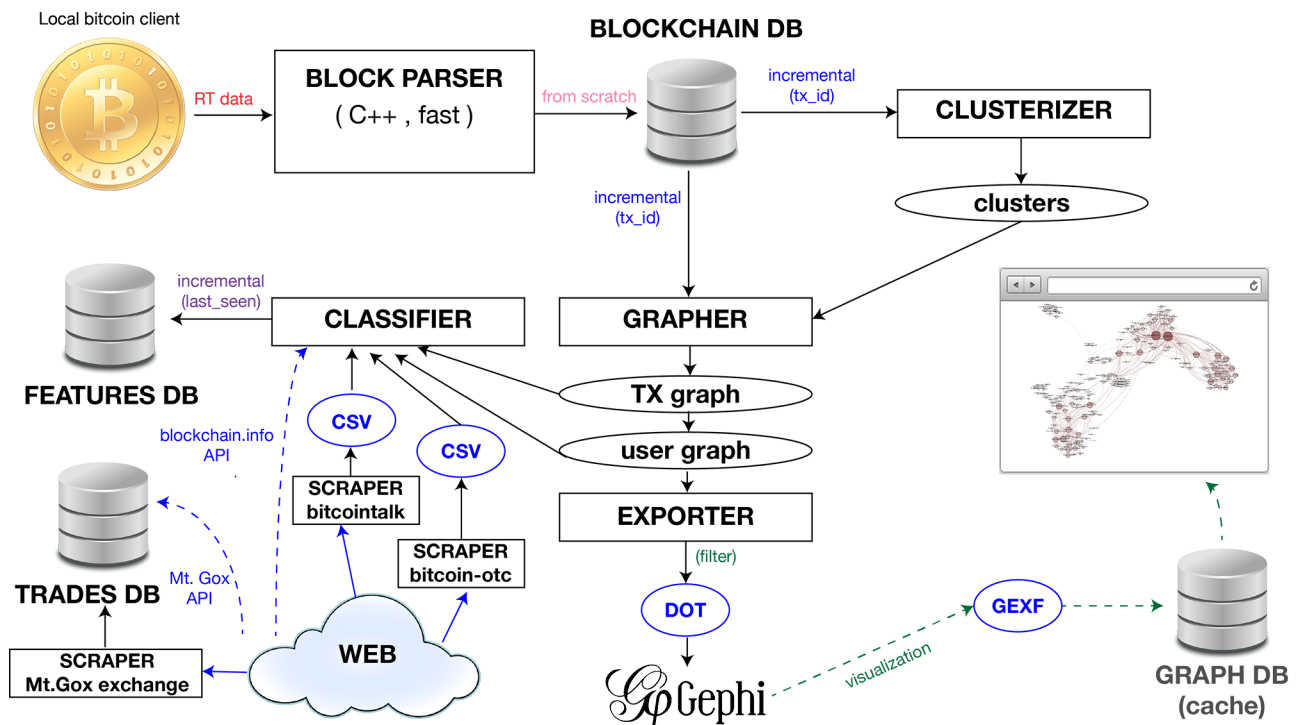


Figure 2: Building blocks of BitIodine

3.2.1 The block parser

The C++ block parser is a modified version of the `blockparser`¹ tool by znort987. The tool reads from the local bitcoin folder populated by the official client and it is a fast way to get updated data from the network. We added several custom callbacks (for example, to highly efficiently export all addresses on the network, to perform a taint analysis on an address, and to export to SQLite).

The custom SQLdump callback exports the blockchain to the *blockchain DB* using a proprietary relational schema we designed in order to obtain good performance when queried by other modules.

The generation of the DB is from scratch, every time, and takes a few minutes on a 2.7GHz Intel i7 CPU with 16GB of RAM.

3.2.2 The clusterizer

The clusterizer, written in Python, incrementally reads the blockchain DB and generates clusters of addresses (*users*) using two heuristics, that will be detailed in the next chapter.

Results are updated incrementally and are saved in a file. It is possible to suspend the generation of clusters, and resume it at any time from where the computation was stopped.

It takes less than 10 minutes to process the whole blockchain, using the same machine.

¹<http://github.com/znort987/blockparser>

3.2.3 The classifier

The Python classifier reads the *transaction graph* and the *user graph* generated by the *grapher* and queries the `blockchain.info`¹ API to get the latest transactions. The classifier has to fill the gap from the most recent transaction saved locally in the graph to the so called *horizon*, which means the meeting point between *real-time* and the state in the graph (frozen at the last run of the *grapher*, which is in turn dependent on the *blockchain DB*).

The result is a set of labels and features per address and/or per cluster, and it is saved in a separate SQLite database for future immediate consultation.

Results are updated incrementally, using the *last seen* timestamp of addresses from the web API.

3.2.4 The scrapers

Python scrapers crawl the web for bitcoin addresses to be associated to real users, generate lists and keep them up to date. Users may be recognized as specific *BitcoinTalk* or *Bitcoin-OTC* users, thanks to crawling of forum signatures and databases.

The interface is easily expandable, and adding scrapers for new services and websites is trivial.

¹<http://blockchain.info>

3.2.5 The Mt. Gox scraper

A Python Mt.Gox scraper uses Mt. Gox trading APIs to get historic data about trades of Bitcoin for US dollars, and saves them in a SQLite database called *trades DB*.

This module is useful to detect interesting flows of coins that enter and exit the Bitcoin economy.

3.2.6 The grapher

The Python grapher incrementally reads the *blockchain DB* and the *cluster file* to generate, respectively, a *transaction graph* and an *user graph*, in two files in NetworkX format.

3.2.7 The exporters

Exporters, in Python, export, eventually filtering, the *transaction graph* and the *user graph*, in DOT format, to be fed into Gephi for visualization.

3.3 Transaction and User graphs

In order to analyze several transactions between seemingly meaningless addresses, it is important to build a graph of them.

In *transaction graphs*, *nodes* are *addresses* and *edges* are single *transactions* between them.

For instance, in the next page is a transaction graph of some activity related to a Bitcoin *faucet* (a website that gives away a very little amount of bitcoins to any visitor that inputs an address). The address of the faucet is 15Art Other important addresses in the graph are 1sLiM . . . , which is owned by BitcoinTalk user *IXIslimIXI*, and 14wQQ . . . , which, as explained later when presenting the tool, is classified by BitIodine as an empty, old address, used for gambling.

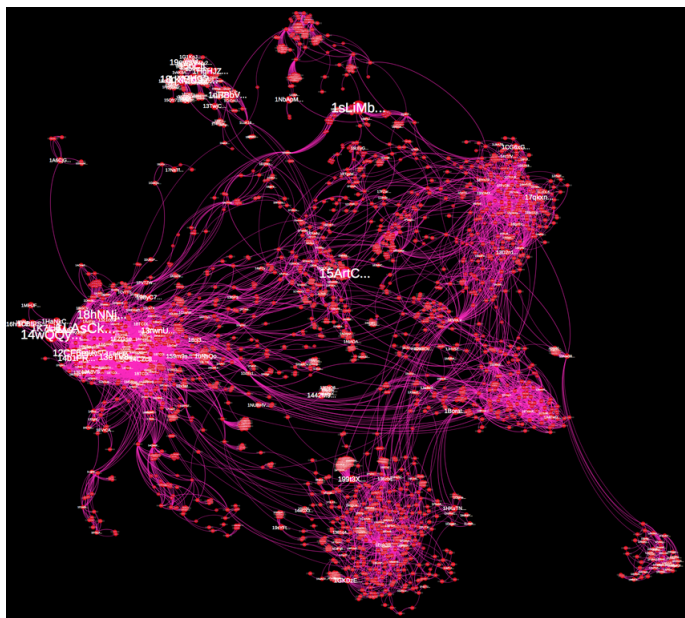


Figure 3: A transaction graph – transactions related to a Bitcoin faucet

BitIodine de-anonymizes addresses by grouping them into *users* or *clusters*.

This is to create an *ownership graph*, compacting the huge transaction graph and giving a new meaning to transactions: edges connecting users are aggregate transactions. They abstract away the complexity of apparently anonymous transactions between meaningless addresses into money exchange between individuals, or entities.

Two heuristics are used to accomplish this.

The first heuristic exploits the fact that when a transaction has multiple input addresses, we can safely assume that those addresses belong to the same wallet, thus to

the same user. This is true under the assumption that owners do not share private keys. This is actually not always true: for example, web wallets have pools that would be mistakenly grouped as a single user.

The second heuristic is about change in transactions. Since the entire value of an unspent output of a prior transaction must be spent and used as input for a new transaction, input is destroyed, and change should be sent back to the user. In order to improve anonymity, a *shadow address* is automatically created and used to collect back the change that results from any transaction issued by the user. The heuristic tries to predict which one of the output addresses is actually belonging to the same user that initiated the transaction, and it does that in two ways: the first one is completely deterministic, the second one exploits a still unfixed flaw in the official bitcoin client that will be detailed in the next chapter.

Using the information obtained by the process, it is possible to group addresses owned by the same individual or entity in macro-nodes that we call, for simplicity, *users*.

In *user graphs*, *nodes* are *users* and *edges* are *macro-transactions* (i.e., flows of coins) between them.

In the following graph, big red nodes are *users*, big blue nodes are addresses of SatoshiDice, a popular Bitcoin gambling site, and the rest are normal, ungrouped addresses.

By just looking at the graph, we can easily spot big players, and the *satellite* structure generated by gamblers moving coins to other addresses not known to be owned by the same person.

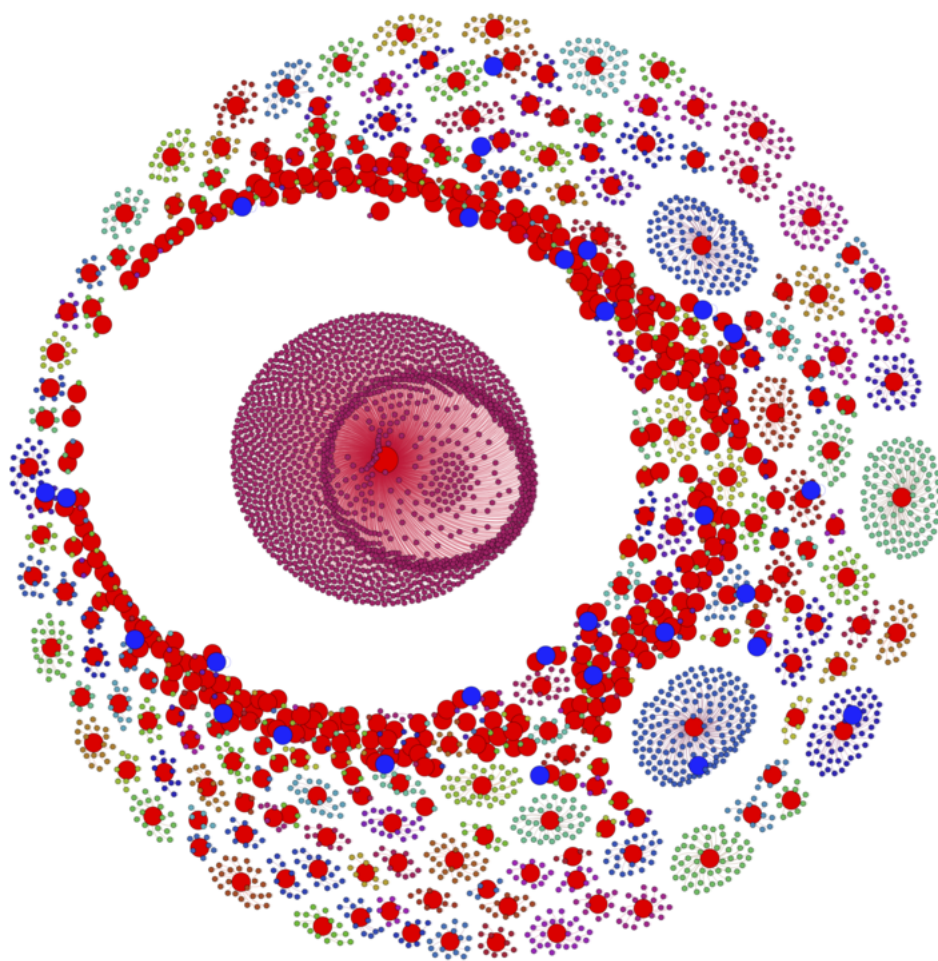


Figure 4: A graph with addresses grouped in users – SatoshiDice gamblers

CHAPTER 4

IMPLEMENTATION

In this chapter we show the code structure of the current implementation of BitIodine, formalize the terminology and definitions we informally presented in Chapter 3, detail the implementation aspects of the heuristics used, evaluate them, justifying design choices and discussing technical aspects.

We finally detail how the *profiling* of the network activity works, providing an example of advanced use of the *Classifier* and the *Exporter* blocks.

4.1 Code structure

This is a high-level overview of the actual BitIodine file structure in the current implementation. Libraries and utility modules for Python modules are not included here for simplicity.

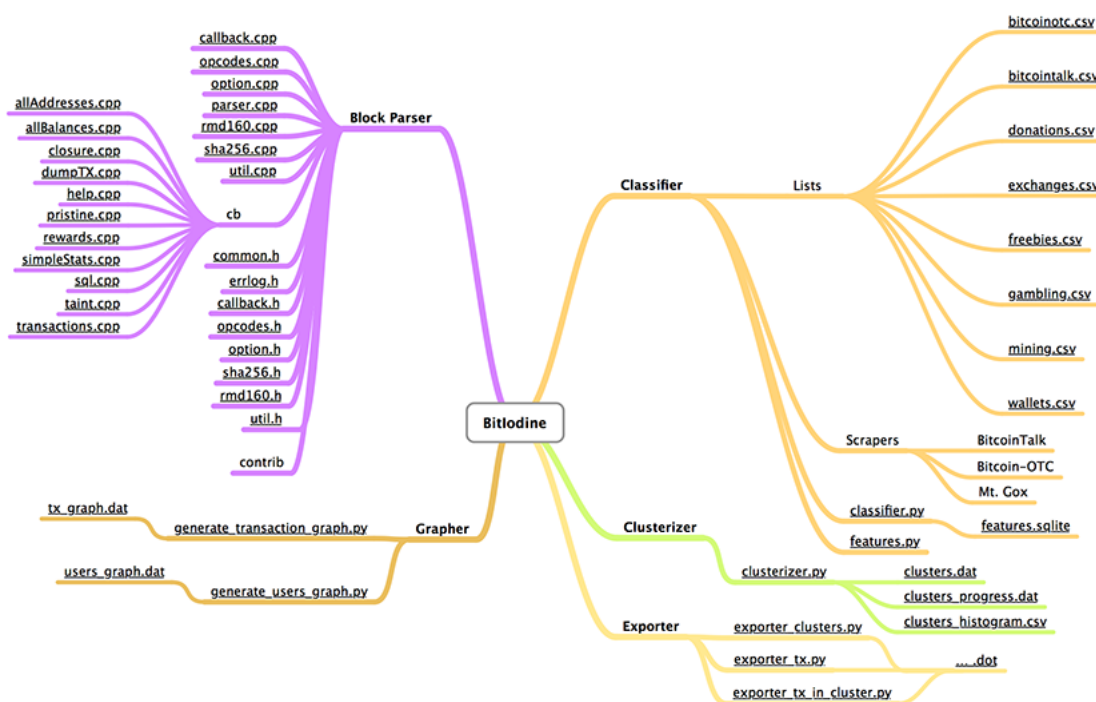


Figure 5: Code structure of BitIodine

4.2 Building the User Graph

The *User Graph* is an important concept in BitIodine.

Compacting the huge transaction graph gives a new meaning to transactions: edges connecting users are aggregate transactions. They abstract away the complexity of apparently anonymous transactions between meaningless addresses into money exchange between individuals, or entities.

Building the *User Graph* in the most accurate possible way is thus crucial. We decided to combine two heuristics, which will be detailed and analyzed below.

4.2.1 A formal model

Here are some definitions that will be used to present the heuristics formally.

Definition 1 (Bitcoin network). *Let N denote the Bitcoin network.*

Definition 2 (Blocks). *Let $B = \{b_1, b_2, \dots, b_{n_B}\}$ be the set of n_B blocks in the Bitcoin network N .*

Definition 3 (Users). *Let $U = \{u_1, u_2, \dots, u_{n_U}\}$ be the set of n_U users in the Bitcoin network N .*

Definition 4 (Addresses). *Let $A = \{a_1, a_2, \dots, a_{n_A}\}$ be the set of n_A addresses in the Bitcoin network N .*

Definition 5 (Transactions). *We assume that within Δt , n_T transactions have taken place as follows:*

$$T = \{\tau_1(S_1 \rightarrow R_1), \tau_2(S_2 \rightarrow R_2), \dots, \tau_{n_T}(S_{n_T} \rightarrow R_{n_T})\}$$

where $\tau_i(S_i \rightarrow R_i)$ denotes a transaction with a unique index i and S_i and R_i denote the sets of senders' addresses and recipients' addresses, respectively.

Definition 6 (Last-block function). Let $lastblock : (T \rightarrow B)$ be a function that maps the set of transactions to the set of blocks.

$lastblock(\tau_i) = b_i$ if and only if b_i is the last block relayed by the network N as the transaction τ_i is broadcast.

Definition 7 (Transactions, limited to a block). $T|_{b_i}$ is defined as a subset of T , containing all the transactions contained in blocks with index $k \leq i$.

Definition 8 (Ownership function). Let $owns : (A \rightarrow U)$ be a function that maps the set of addresses to the set of users.

$owns(a_i) = u_k$ if and only if u_k owns the private key of a_i .

4.2.2 The first heuristic: Multi-input transactions grouping

Definition 9 (Multi-input transaction). Multi-input transactions occur when an user u wishes to perform a payment, and the payment amount exceeds the value of each of the available bitcoins in u 's wallet. In order to avoid performing multiple transactions to complete the payment, enduring losses in terms of transaction fees, Bitcoin clients choose a set of bitcoins from u 's wallet such that their aggregate value matches the payment and perform the payment through multi-input transactions.

The first heuristic exploits the fact that when a transaction has multiple input addresses, we can safely assume that those addresses belong to the same wallet, thus to the same user.

More formally:

Definition 10 (First heuristic). *Let*

$$\tau_i(S_i \rightarrow R_i) \in T$$

be a transaction, and

$$S_i = \{a_1, a_2, \dots, a_{n_{S_i}}\}$$

the set of input addresses.

Let also $|S_i| = n_{S_i}$ be the cardinality of the set.

If $n_{S_i} > 1$, then all input addresses belong to the same (previously known or unknown) user:

$$\text{owns}(a_i) \triangleq u_k \quad \forall i \in S_i$$

This is true under the assumption that owners do not share private keys. This is actually not always true: for example, web wallets have pools that would be mistakenly grouped as a single user.

4.2.3 The second heuristic: shadow address guessing

The second heuristic has to do with *change* in transactions.

Since the entire value of an unspent output of a prior transaction must be spent and used as input for a new transaction, input is destroyed, and change shall be sent back to the user.

In order to improve anonymity, a *shadow address* is automatically created and used to collect back the change that results from any transaction issued by the user.

The heuristic tries to predict which one of the output addresses is actually belonging to the same user than initiated the transaction, and it does that in two ways: the first one is completely deterministic, the second one exploits a still unfixed flaw in the official Bitcoin client that will be detailed below, in Section 4.2.3.2.

4.2.3.1 Variant 1 – deterministic version

This variant is completely deterministic, and can be considered the *conservative* variant, since it is guaranteed to work correctly.

In words, if there are two output addresses (this is true for more than 90% of transactions, as we show in Section 4.2.4.2), and one of the two has never appeared before in the blockchain, while the other has, then we can safely assume that the one that never appeared before is the one generated by the client to collect change back, the *shadow address*.

More formally:

Definition 11 (Second heuristic – variant 1). *Let*

$$\tau_i(S_i \rightarrow R_i) \in T$$

be a transaction, and

$$R_i = \{a_1, a_2, \dots, a_{n_{R_i}}\}$$

the set of output addresses.

Let also $|R_i| = n_{R_i}$ be the cardinality of the set, and $T|_{lastblock(\tau_i)}$ the set T limited to the last block at the time of transaction τ_i .

If $n_{R_i} = 2$, then the output addresses are a_1 and a_2 .

If $a_1 \notin T|_{lastblock(\tau_i)}$ and $a_2 \in T|_{lastblock(\tau_i)}$, then a_1 is the shadow address, and belongs to the same user u_k who owns the input address(es):

$$owns(a_1) \triangleq u_k$$

4.2.3.2 Variant 2 – exploiting a logical flaw in the official bitcoin client

In the Bitcoin official client, in file `src/wallet.cpp`¹, we can see that when the client chooses in which slot to put the shadow address, it passes to `GetRandInt` the number of payees. In the common case of one payee, `GetRandInt` will always return 0, and the change always ends up in the first output.

```
// Insert change txn at random position

int list_begin = wtxNew.vout.begin();

int n_of_payees = wtxNew.vout.size();
```

¹<https://github.com/bitcoin/bitcoin/blob/master/src/wallet.cpp>

```
vector<CTxOut>::iterator position = list_begin + GetRandInt(n_of_payees);
wtxNew.vout.insert(position, CTxOut(nChange, scriptChange));
```

It is an extremely common off-by-one error, and the code should be corrected as follows:

```
// Insert change txn at random position
int list_begin = wtxNew.vout.begin();
int n_of_payees = wtxNew.vout.size() + 1;
vector<CTxOut>::iterator position = list_begin + GetRandInt(n_of_payees);
wtxNew.vout.insert(position, CTxOut(nChange, scriptChange));
```

In Section 4.2.4.2, we demonstrate that just 6.8% of transactions have the shadow address provably in the second, last, slot of two-outputs transactions.

This gives us confidence to relax the second heuristic and just check for the deterministic cases (one of the two output addresses has never appeared before in the blockchain, while the other has) and also consider a previously unseen first output in two-outputs transactions as a shadow address, without checking for the other one.

This allows for a much better coverage, and generates much more compact clusters of users, as showed in Figure 6.

4.2.4 Evaluating heuristics

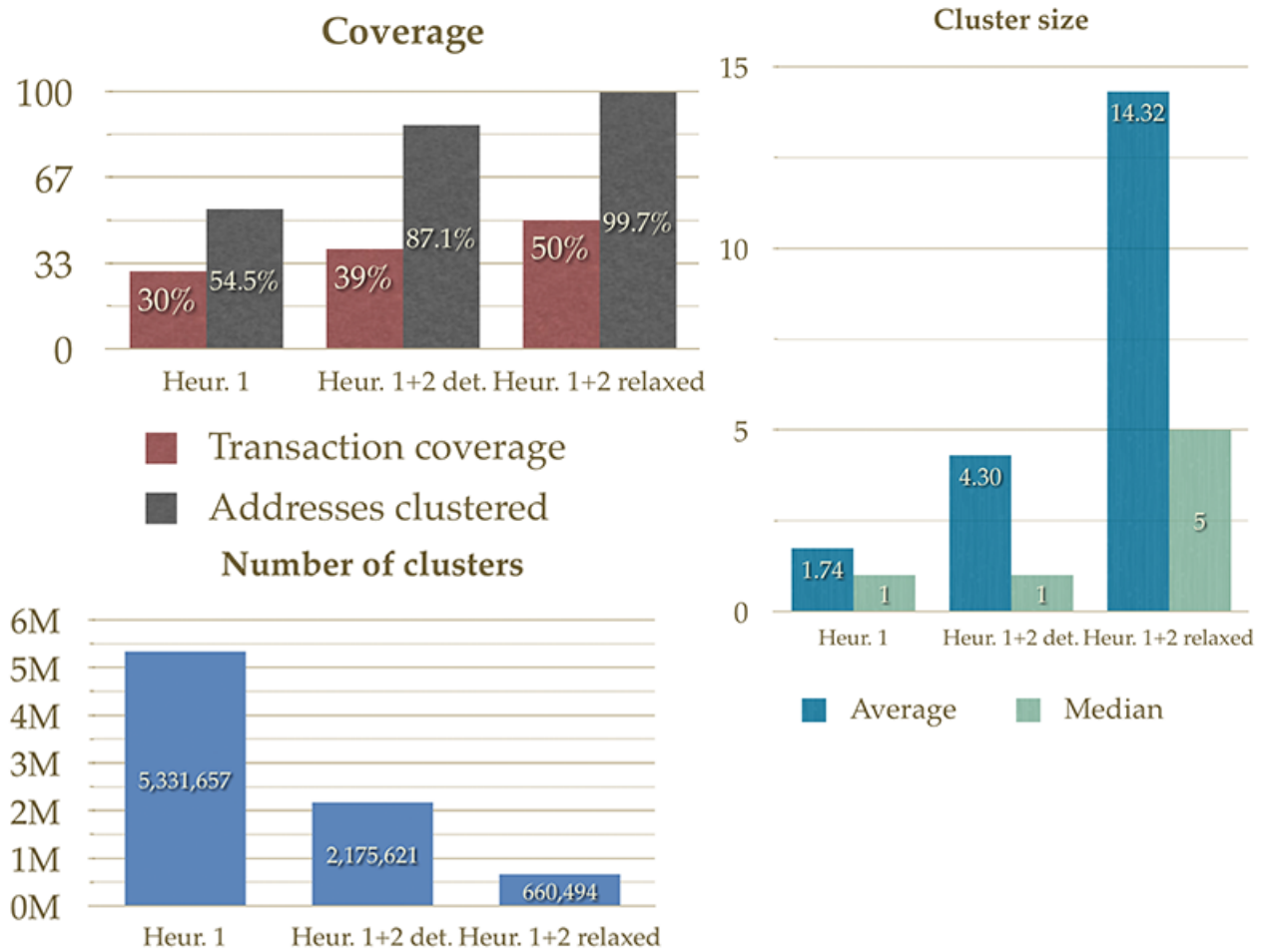


Figure 6: Statistics about clusters while evaluating heuristics

4.2.4.1 First heuristic

Using just the first heuristic, 30% of transactions contribute to building clusters (Table I). This means that approximately one third of transactions have multiple input addresses.

Very sparse clusters are generated, where most of clusters are actually just made of a single address (*singletons*). The biggest cluster is made of 1673 addresses, and the average cluster size is 1.74 (Table II).

4.2.4.2 Second heuristic

First of all, as we can see in Figure 7, more than 90% of transactions have exactly two outputs, and carry the great majority of the volume (Figure 8). This is because most transactions have just one payee, and one shadow address to collect change back.

This gave us confidence to combine the first heuristic to the second one, in two variants.

Using the first heuristic combined with the second in the first, deterministic, variant, 38.8% of transactions contribute to building clusters (Table I).

TABLE I: TRANSACTION COVERAGE OF DIFFERENT HEURISTICS

	Heuristic 1	Heur. 1+2 (deterministic)	Heur. 1+2 (variant 2)
Transactions	3960331 / 13069469	5075743 / 13069469	6537279 / 13069469
Coverage	30%	38.8%	50%

A smaller number of more compact clusters are generated, as we can see in Figure 6. The biggest cluster is made of 62898 addresses, and the average cluster size is 4.3 (Table III).

Python code of the algorithm is in Appendix C.

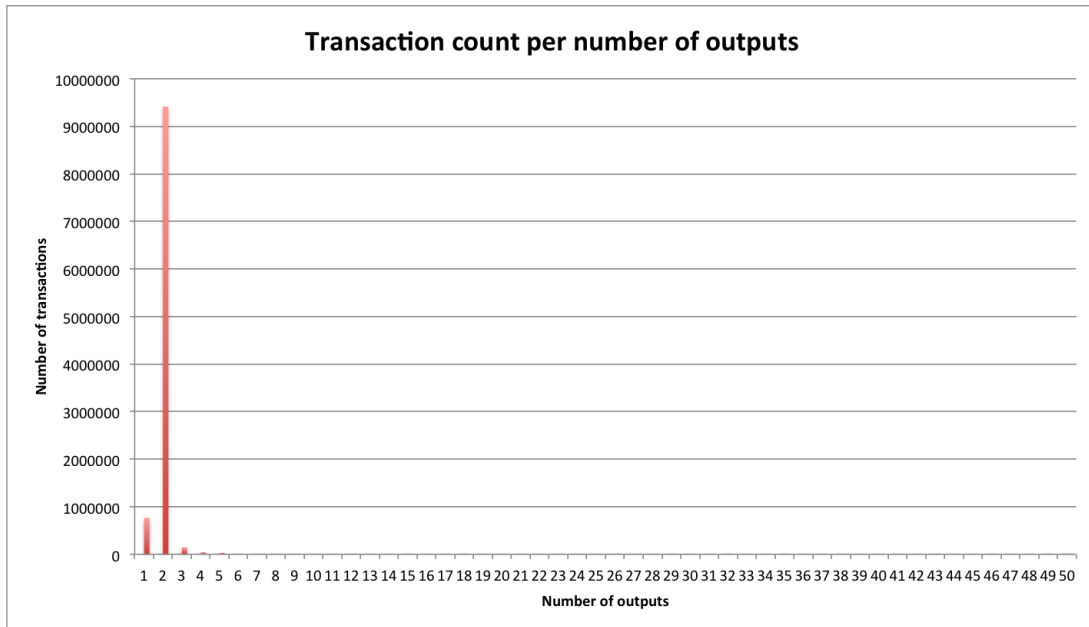


Figure 7: Transaction count per number of outputs

TABLE II: CLUSTERING WITH FIRST HEURISTIC ONLY

Number of clusters	Min. size	Average size	Median size	Max. size
5331657	1	1.74	1	1673

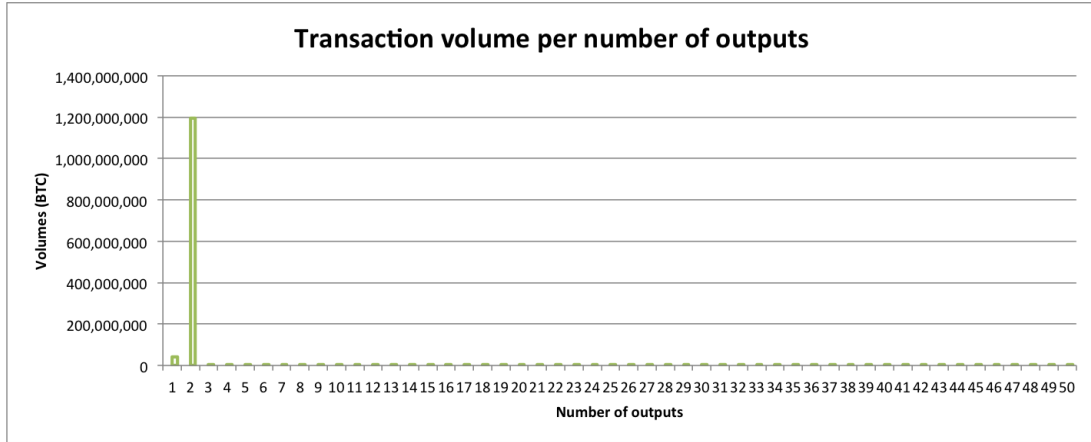


Figure 8: Transaction volume per number of outputs

TABLE III: CLUSTERING WITH HEURISTICS 1 AND 2 (DETERMINISTIC)

Number of clusters	Min. size	Average size	Median size	Max. size
2175621	1	4.3	1	62898

In order to decide whether it is convenient to relax the second heuristic and just check for the deterministic cases (one of the two output addresses has never appeared before in

the blockchain, while the other has) and also consider a previously unseen first output in two-outputs transactions as a shadow address, without checking for the other one, we decided to look for an estimate of how many transactions have the shadow address provably in the second, last, slot of two-outputs transactions, and so are *not* relayed by the official Bitcoin client, which has the logical flaw.

Of the 5075743 transactions that contributed to form clusters (Table I), 4730425 (93.2%) have a guaranteed shadow address in the first slot, and only 345318 (6.8%) in the last slot. 1806854 (13.8% of the total number of transactions) have both output addresses unseen, and can be considered our margin of error.

We then decided to exploit the client bug, finally getting a much smaller number of much more compact clusters, as we can see in Figure 6. The biggest cluster is made of 104248 addresses, and the average cluster size is 14.32 (Table IV).

TABLE IV: CLUSTERING WITH HEURISTICS 1 AND 2

Number of clusters	Min. size	Average size	Median size	Max. size
660494	1	14.32	5	104248

4.3 Employed technologies

4.3.1 SQLite databases

We opted for the use of embedded SQLite databases for storing the blockchain and the features database because it is a zero-configuration, server-less, embedded, stable and compact cross-platform solution.

We do not need concurrency while writing to database files, so the only possible disadvantage does not affect its use in BitIodine.

4.3.1.1 Blockchain DB schema

In Figure 9 is the relational schema we designed to contain the blockchain. As mentioned earlier, the blockchain DB is populated by the *block parser* module.

The full SQL schema can be found in Appendix B.

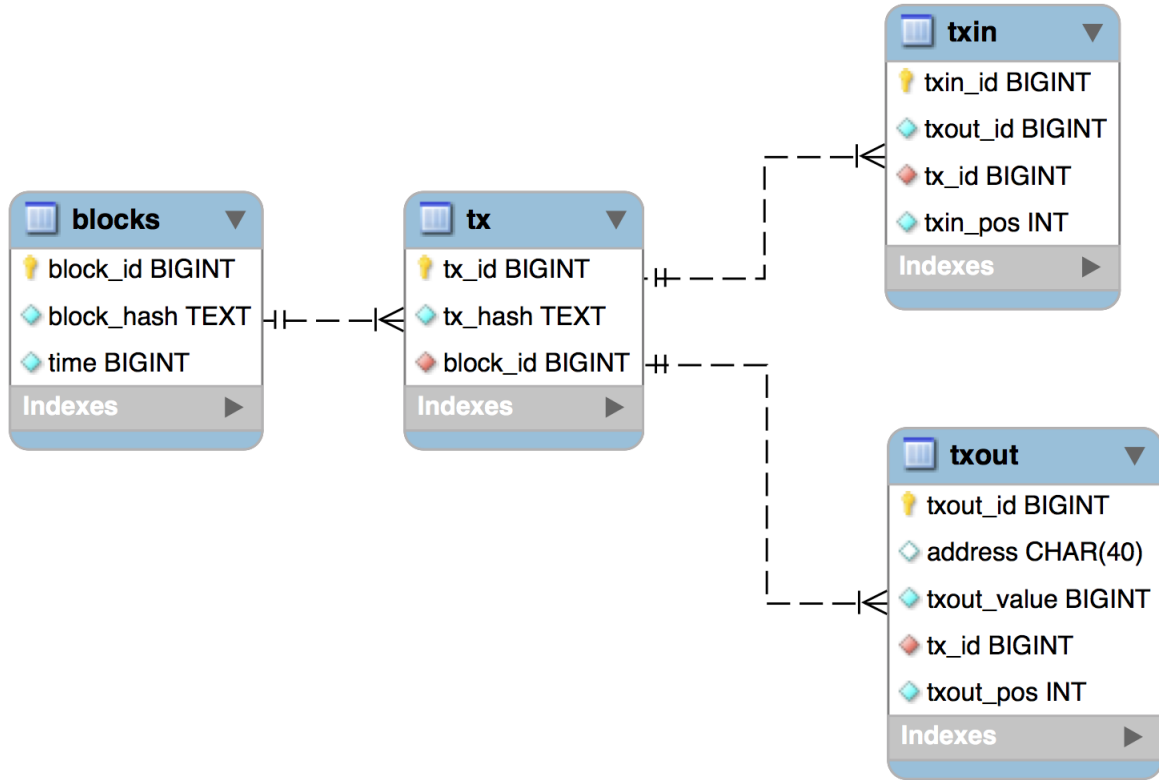


Figure 9: SQL schema diagram for the blockchain representation

4.3.1.2 Features DB schema

In Figure 10 is the relational schema we designed to store features of classified addresses and clusters. This database is populated by the *classifier* module.

The full SQL schema can be found in Appendix B.

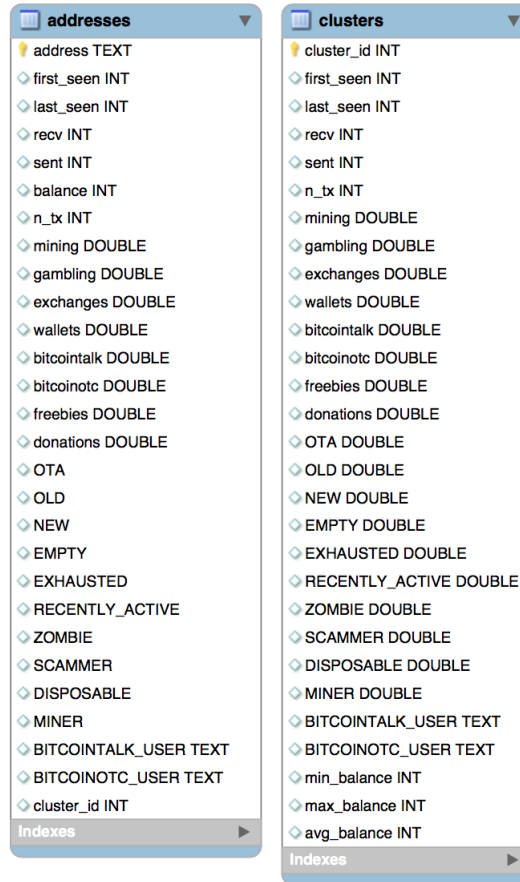


Figure 10: SQL schema diagram for the features DB

4.3.1.3 Trades DB schema

In Figure 11 is the relational schema we designed to store trades in the Mt. Gox exchange. This database is populated by the *Mt. Gox scraper* module.

The full SQL schema can be found in Appendix B.

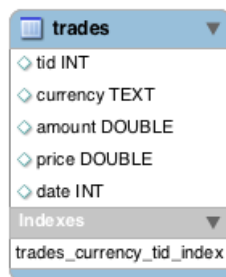


Figure 11: SQL schema diagram for the trades DB

4.3.2 NetworkX graphs

We decided to use NetworkX, a Python language software package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks¹, for the internal representation of users and transactions graphs.

NetworkX objects can be serialized (in Python jargon, *pickled*) and written to a file with ease, and querying for successors and predecessors of nodes is very efficient. Is it also possible to embed an arbitrary number of additional data labels to nodes and edges (we added the number of transactions between two addresses, for example).

4.4 Using the Classifier

The *Classifier* exposes a command-line interface, and can be directly used by the end user.

¹<http://networkx.github.io>

Here is the *help*:

```
./classifier.py --help
usage: classifier.py [-h] [-d DB] [-a ADDRESS] [-c CLUSTER] [--all-clusters]

BitIodine Classifier

optional arguments:
  -h, --help            show this help message and exit
  -d DB                 SQLite database path
  -a ADDRESS            Classify a single address.
  -c CLUSTER            Classify a single cluster.
  --all-clusters        Classify every cluster.
```

As we can see, it is possible to classify *addresses* and whole *clusters (users)*.

When we choose to classify a single address, the Python classifier reads the *transaction graph* and the *user graph* generated by the *grapher* and queries the `blockchain.info`¹ API to get the latest transactions. The classifier has to fill the gap from the most recent saved locally in the graph to the so called *horizon*, which means the meeting point between *real-time* and the state in the graph (frozen at the last run of the *grapher*, which is in turn dependent on the *blockchain DB*).

The result is a set of labels and features per address, and it is saved in the *features database* for future immediate consultation.

When we ask the classifier to classify an *user*, every address belonging to that user is classified, and boolean flags (*True / False*) are averaged, thus becoming a real number (for example, 0.5 means that half of the addresses in that cluster have that flag set).

¹<http://blockchain.info>

Labels for addresses and clusters, with their description, are in Appendix A.

4.4.1 Exporting and visualizing

BitIodine also allows for plotting of graphs and sub-graphs in Gephi¹.

Since Gephi can not visualize NetworkX graphs directly, *exporter* modules take care of the conversion process in the text-based DOT format, and also allow the user to filter with various criteria: minimum amount of BTC sent by *a* to *b* to have an edge between them, minimum number of transactions between nodes, min/max in/out degree of nodes.

It is also possible to export just a single cluster, generating an *hybrid graph*, where nodes can be either *addresses* (not in a non-singleton cluster) or *users*. An example is represented in Figure 12.

¹<http://gephi.org>



Figure 12: Hybrid graph of transactions of more than 100 BTC

CHAPTER 5

EXPERIMENTAL RESULTS

In this chapter we present some use cases for BitIodine, and we demonstrate that we are able to extract intelligence from the network activity, identifying an address and binding it to a BitcoinTalk username and a Bitcoin-OTC trader and *scammer*.

Finally, we demonstrate that BitIodine is able to demonstrate that an address belongs to the Silk Road, and we analyze the activity surrounding a mining pool hack and the famous 2.5 million dollars pizza purchase in May 2010 we outlined in Chapter 2.

5.1 Use cases: classifying addresses and users

In this first use case, we demonstrate that we are able to extract intelligence from the network activity, identifying an address we found in IRC chat logs of channel `#bitcoin-otc`¹, an over-the-counter marketplace for trading with Bitcoin: `1MisakifjKcG2KWKgY4XES1bkAxdU4ozSE`.

First, we ask the *classifier* to provide information about the address itself, as we can see in Figure 13.

We can see in Table V that it is a zero-balance address, exhausted in 84 transactions, belonging to user *xisalty* on BitcoinTalk forum and user *xisalty-otc* on the exchange Bitcoin-OTC.

BitIodine informs us that that the address has been marked as belonging to a *scammer*. This is because the *Bitcoin-OTC scraper* module automatically retrieved the ratings of the user and decided that the user received a significative amount of *negative* ratings from trustworthy users.

In fact, a quick Google search confirms that the user is a clone of a known scammer (*SupaDupaJenkins*, aka *SupaDupa*, aka *Supa*), and that, for example, (s)he defaulted a loan granted by Bitcoin-OTC user *Hasimir*². The victim of the scam also publishes loan details, IRC chat logs, and GPG signed copies of the loan contract and terms (Figure 14).

¹<http://www.adversary.org/bitcoin-otc/xisalty/xisalty-loan-20121111.txt>

²http://www.adversary.org/wp/?page_id=7297

Now we can get more information about the user itself, by asking the classifier to classify the cluster to which the address belongs (in Table V we can read the cluster ID).

As we can read in Table VI, every address in the cluster is empty, 2.3% of them are *zombies*, which means they have been empty and dormant for a long time, then got used again, and 4.5% are One-Time-Addresses.

```
Miki@MikiMac:~/PoliMi/Laurea Magistrale/Tesi/bitiodine/deploy/classifier$ master ⚡ ret 130 => ./classifier.py -a 1MisakifjKcG2KWKgY4XES1bkAxdU4ozSE
Graph loaded.
Clusters loaded.
Singletons stripped.
Classifying address 1MisakifjKcG2KWKgY4XES1bkAxdU4ozSE...
Address is already up to date. Partial update...
{'BITCOINOTC_USER': 'xisalty-otc',
 'BITCOINTALK_USER': 'xisalty',
 'DISPOSABLE': False,
 'EMPTY': True,
 'EXHAUSTED': True,
 'MINER': False,
 'NEW': False,
 'OLD': False,
 'OTA': False,
 'RECENTLY_ACTIVE': False,
 'SCAMMER': True,
 'ZOMBIE': False,
 'balance': 0,
 'bitcoinctc': 0.014084507042253521,
 'bitcointalk': 0.014084507042253521,
 'cluster_id': 246429,
 'donations': 0.0,
 'exchanges': 0.0,
 'first_seen': 1351893426,
 'freebies': 0.0,
 'gambling': 0.0,
 'last_seen': 1357532709,
 'mining': 0.0,
 'n_tx': 84,
 'recv': 4207555841,
 'sent': 4207555841,
 'wallets': 0.0}
```

Figure 13: Classifying an address with BitIodine

```

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

I, xisalty-otc, agree to the following terms and conditions for taking a
Bitcoin loan from Ben McGinnes (Hasimir) on #bitcoin-otc:

Borrower: xisalty-otc
Borrower GPG key: 0xB8571D279AA089BC
Lender: Ben McGinnes (Hasimir)
Lender GPG key: 0x321E4E2373590E5D

Loan amount: 5 BTC
Loan rate: 50%
Compound Interest: no
Per period: 15 days
Loan details: 5 BTC at 50% per 15 day(s)

Amount Due after 15 days: 7.5 BTC
Amount Due after 22 days: 8.0 BTC

Loan period: 22 days
Extended loan period: 66 days
Penalty rate: add 1% per day

IN PGP SIGNED MESSAGE-----
Message for loan: 1MisakifjKcG2KWKgY4XES1bkAxdU4ozSE
A512               repayment: 1GEYmjLddLZhcAo5aJp4qXp66EDhmm13Cm

the required timeframe the loan
is repaid.

-understa

<gribble> Currently authenticated from hostmask xisalty-
:y@unaffiliated/xisalty . User xIsalty-otc, rated since Wed Nov 7 23:47:04
ulative rating 3, from 3 total ratings. Received ratings: 3 positive, 0
Sent ratings: 3 positive, 1 negative. Details: http://bitcoin-
iewratingdetail.php?nick=xIsalty-otc

salty:

```

Figure 14: GPG signed contract of a defaulted loan, and IRC chat logs

TABLE V: USE CASE: ADDRESS LABELS












Label	Value
cluster_id	246429
first_seen	Fri, 02 Nov 2012 21:57:06 GMT
last_seen	Mon, 07 Jan 2013 04:25:09 GMT
recv	42.07555841 BTC
sent	42.07555841 BTC
balance	0 BTC
n_tx	84
mining	0%
gambling	0%
exchanges	0%
wallets	0%
bitcointalk	1.4%
bitcoinotc	1.4%
freebies	0%
donations	0%
OTA	
OLD	
NEW	
EMPTY	
EXHAUSTED	
RECENTLY_ACTIVE	
ZOMBIE	
SCAMMER	
DISPOSABLE	
MINER	
BITCOINTALK_USER	xisalty
BITCOINOTC_USER	xisalty-otc

TABLE VI: USE CASE: CLUSTER LABELS

Label	Value
cluster_id	246429
first_seen	Fri, 16 Sep 2011 21:38:13 GMT
last_seen	Mon, 07 Jan 2013 04:25:09 GMT
recv	84.93059065 BTC
sent	84.93053224 BTC
min_balance	0 BTC
max_balance	0.00004598 BTC
avg_balance	0.0000013275 BTC
n_tx	168
mining	0%
gambling	0%
exchanges	0%
wallets	0%
bitcointalk	2.3%
bitcoinotc	2.3%
freebies	0%
donations	0%
OTA	4.5%
OLD	2.3%
NEW	0%
EMPTY	100%
EXHAUSTED	95.5%
RECENTLY_ACTIVE	0%
ZOMBIE	2.3%
DISPOSABLE	0%
MINER	0%
BITCOINTALK_USER	xisalty
BITCOINOTC_USER	xisalty-otc
SCAMMER	

5.1.1 A real-world case: investigating the Silk Road

We would like to identify the addresses owned by the Silk Road, the large black market that makes use of Bitcoin we talked about in Chapter 2.

We can use the *block parser* module to display the top N addresses ordered by balance passing balances -l N to it.

One of the addresses that moved most funds on the network in 2012 is 1DkyBEKt5S2GDtv7aQw6rQepAvnsRyHoYM.

As we can see in Figure 15, it is relatively recent, and has been accumulating coins steadily since April 2012, becoming completely empty in September of the same year. In total, the address has received 613,326 BTC. The address belongs to a cluster of 7 addresses, most of them input of very large transactions.

By analyzing the activity of the address with the *block parser* module and by querying the *trades DB* populated by the *Mt. Gox scraper*, we find that:

- On July 17, 2012 this address had a balance of 517,825 BTC.
- On the same day Bitcoin looked on the verge of breaking 10 USD/BTC on Mt. Gox.
- At 02:00 AM, someone sold 10,000 BTC at 9 USD/BTC, driving the price down. This is indicated in Figure 16 with a red arrow.

- At 02:29, two large withdrawals of respectively 20,000 BTC¹ and 60,000 BTC² were made from this address one after the other (11 seconds between them), and included in a block at 02:32.
- Mt. Gox, at the time, needed 6 confirmations in order to allow the user to spend deposited bitcoins. 6 confirmations were matured with block at height 189421, relayed at 02:47:24 AM.
- A few minutes after that, at 02:52 and 02:53, someone sold approximately 15,000 BTC at market price in several batches (the only two trades for more than 1,000 BTC are shown in Figure 17), causing the price to drop below 7.5 USD/BTC. This is highlighted in Figure 16 with a yellow background.

¹<https://blockchain.info/tx/3cb4f452fd0e5719391a6f1b82cd80a329a86734350ca04cab81d0e94f94e709>

²<https://blockchain.info/tx/cea22747487e8a2824566fa362981782871fea50fdfb690a3b63d85bd3189593>

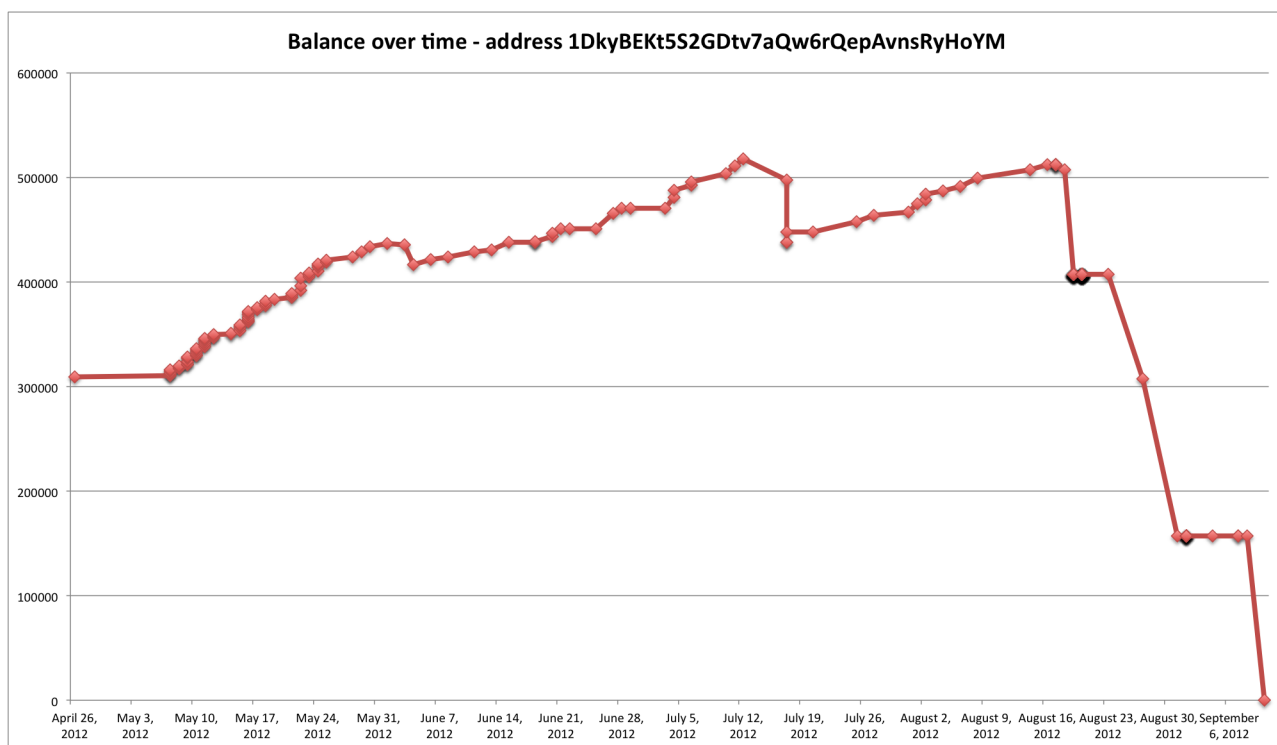


Figure 15: Plot of balance over time of a Silk Road-owned address



Figure 16: Bitcoin price chart for 17 Jul 2012

We can speculate that the rich owner of so many bitcoins was scared by the move down in price (or, (s)he caused that with existing funds on the exchange), and sold a bunch at market price.

Thanks to our tool, we are also able to prove that the address is actually owned by the Silk Road.

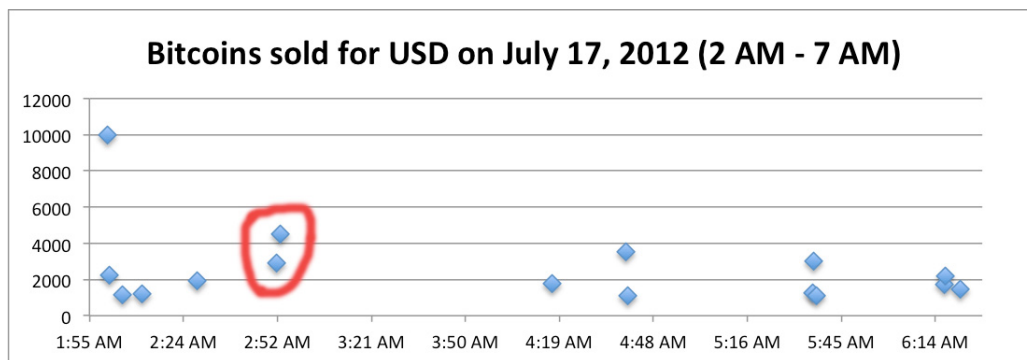


Figure 17: Two big trades on the Mt. Gox exchange

A BitcoinTalk user, on July 29, 2012, posted an important piece of information: the fact he deposited 0.001 BTC to his Silk Road account, and the address he was given as a deposit address.

According to the *Clusterizer*, the deposit address (1Q6nyjSQ79AAw67xAGHgXxXHRj9erLLqhD) is provably in the same wallet as more than 25,000 other addresses. This is because the

Silk Road scrambles addresses, mixing funds and splitting them in thousands of one-time addresses in order to make investigations more difficult.

We can use the list of addresses in the cluster to find whether there was any connection between these addresses and the large 1Dky... address.

The cluster is active since June 18, 2012, and there have been more than 80,000 inputs and outputs to/from these addresses. Since we do not see older transactions in the list, it seems the Silk Road wallet must have been reset around June 18, 2012.

By following the flow of coins, we notice that the 0.001 BTC is being grouped with a few other Silk Road-owned addresses¹.

In the transaction in Figure 18, we notice that the deposit address and 1AVM... are inputs to the same transaction. In fact, our huge cluster of more than 25,000 addresses confirms that both the deposit address and 1AVM... are in the same wallet.

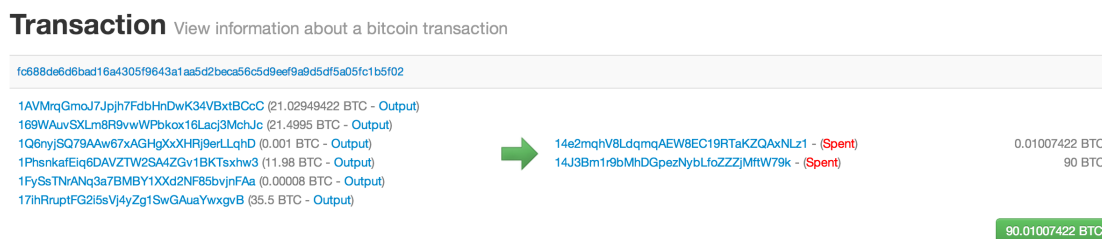


Figure 18: An important transaction in our Silk Road investigation

¹<https://blockchain.info/tx/fc688de6d6bad16a4305f9643a1aa5d2beca56c5d9eef9a9d5df5a05fc1b5f02>

As we can see in Figure 19, a big multi-input payment to our address 1Dky... has 1AVM... as one of the payers¹ – this means we have spotted a multi-hop connection from our deposit address to the big address we were wondering whether it belonged to Silk Road.



Figure 19: The transaction that links the address to Silk Road

Also, we can spot other addresses in the same cluster as input addresses to 1Dky...

Thanks to our tools, we have concluded that the large address is indeed related to the Silk Road.

¹<http://blockchain.info/tx/7c43eba80f90c770b8a5e3d196df7138fadbc62b2c81fc234fa48023bc23a8b2>

5.1.2 Another real-world case: the Slush mining pool hack

On March 1, 2012, a very popular mining pool operated by a respected BitcoinTalk member, *slush*, got hacked through a vulnerability in Linode Manager, the control panel of the famous VPS provider Linode¹.

3094 BTC *hot coins* (i.e., ready for payouts) were transferred to address 1NRy8GbX56MymBhDYMyqsNKwW9VupqKVG7 in transaction 34b84108a142ad7b6c36f0f3549a3e83dcdbb60e0ba0df96cd48f852da0b1acb.

Using the *Clusterizer* to generate a list of addresses in the cluster where the 1NRy... address is, and the *Block Parser*, it is possible to quickly dump the transactions in which addresses belonging to the thief appear (see Appendix D).

In particular, an investigation can focus on pre-theft activity by the thief. In this case, we can see that thousands of bitcoins are moved between addresses owned by the thief before the hack.

We obtain similar results with the *Classifier*, which confirms that after 21 March 2012, at 11:33:13 GMT, the money leaves the known cluster and the balance is zero.

The *Classifier* also highlights that every address in the cluster is now empty.

¹<https://www.linode.com>

5.1.3 An expensive pizza

In May of 2010, a BitcoinTalk user called *laszlo* from Jacksonville, Florida, bought two pizzas for 10,000 BTC¹. Another user, *jercos*, bought the two pizzas to be delivered to him and posted photos as proof².

10,000 BTC were valued \$41 at the time of the trade. In April 2013, they can be sold for 2.5 million USD.

The transaction ID is public (a1075db55d416d3ca199f55b6084e2115b9345e16c5cf302fc80e9d5fbf5d48d), so we ask the *Classifier* to profile the cluster to which the input address belongs.

The *Classifier* is indeed able to identify the BitcoinTalk user (*laszlo*), and we also get more information about the user: the cluster is zero-balance, all the addresses are old and the last address used was on 20 Aug 2012. 89,211 BTC were received and sent by addresses in the cluster.

Furthermore, we can get more insight about the nature of the addresses used to perform the payments. About half of them are *mining addresses*, and they all got 50 BTC rewards. In Figure 20 we can see in blue mining addresses, in red other addresses, and transactions between them as edges.

¹<https://bitcointalk.org/index.php?topic=137.0>

²<http://heliacal.net/~solar/bitcoin/pizza/>

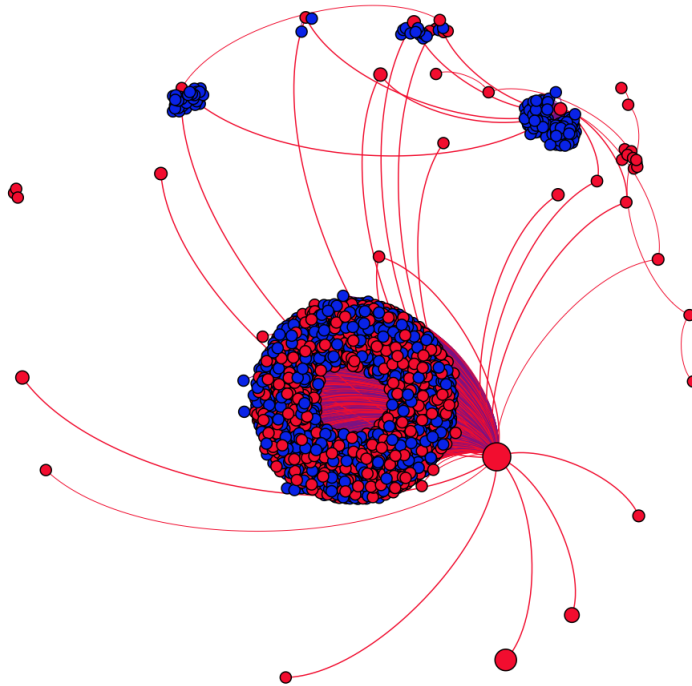


Figure 20: Graph of transactions inside the *laszlo* cluster

Most mining addresses rewards are spent directly, some are first transferred to a big red *hub* (with address 1XPTg...) and the majority of transactions is between other addresses.

This means that, back in 2010, *laszlo* was a miner that decided to give out some bitcoins to BitcoinTalk forum members.

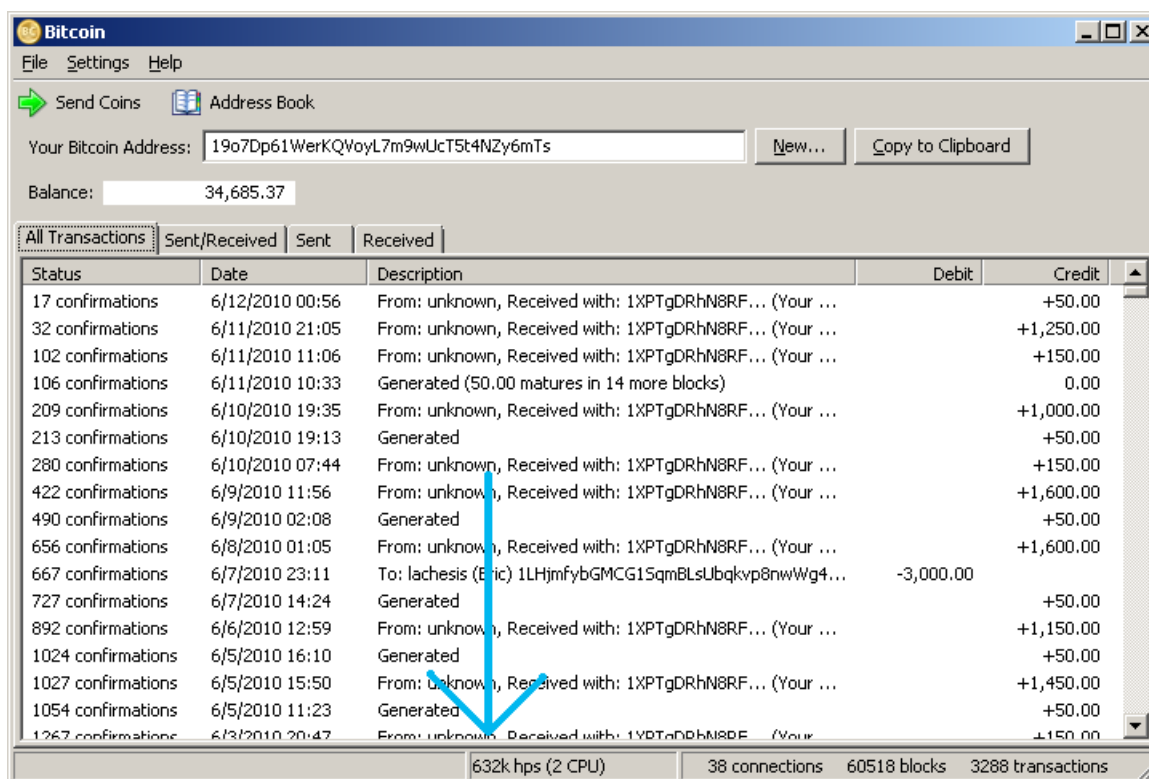


Figure 21: Screenshot posted by *lazzlo* of his wallet

This is confirmed by a screenshot that *lazzlo* publicly posted on the forum (Figure 21), in which we can see the mining activity (the 50 BTC *Generated* transactions) and the big hub found by BitIodine (with address 1XPTg...).

CHAPTER 6

CONCLUSIONS

In this thesis we presented BitIodine, a collection of modules to parse the blockchain, cluster addresses, classify addresses and users, graph, export and visualize elaborated information from the Bitcoin network.

We tested our tool on previously known and unknown use cases, and we demonstrated that we were able to get valuable information. In particular, we showed that using a combination of modules it is possible to prove that one bitcoin address actually belongs to the Silk Road, the large black market.

Since the BitIodine architecture is modular, expandable and easily deployable, we expect BitIodine to be used in the future as a skeleton for building more complex frameworks for Bitcoin forensic analysis.

There is room for future improvement – next step will be adding a Neo4j Enterprise graph database and a caching solution for easily visualizing graphs in a web application, in order to add an user-friendly frontend to the framework.

APPENDICES

Appendix A

LABELS FOR ADDRESSES AND CLUSTERS

A.1 Labels for addresses

In Table VII we list the labels for addresses with their description.

Appendix A (Continued)

TABLE VII: LABELS FOR ADDRESSES

Label	Type	Meaning
first_seen	NUMBER	Timestamp of the first appearance of the address in a transaction
last_seen	NUMBER	Timestamp of the last appearance of the address in a transaction
recv	NUMBER	Total amount received by the address
sent	NUMBER	Total amount sent from the address
balance	NUMBER	Balance of the address
n_tx	NUMBER	Number of transactions in which the address appears
cluster_id	NUMBER	The ID of the cluster the address belongs to
mining	RATIO	Ratio of transactions coming from direct or pooled mining
gambling	RATIO	Ratio of transactions to/from gambling sites
exchanges	RATIO	Ratio of transactions to/from exchanges
wallets	RATIO	Ratio of transactions to/from web wallets
bitcointalk	RATIO	Ratio of transactions to/from known BitcoinTalk users
bitcoinotc	RATIO	Ratio of transactions to/from known Bitcoin-OTC users
freebies	RATIO	Ratio of transactions to/from faucets or other freebies
donations	RATIO	Ratio of transactions to/from known donation addresses
OTA	BOOLEAN	One-Time-Address: appears in just one transaction
OLD	BOOLEAN	Not seen for a long time (<i>tunable</i>)
NEW	BOOLEAN	First appearance is recent (<i>tunable</i>)
EMPTY	BOOLEAN	Balance is close to zero
EXHAUSTED	BOOLEAN	EMPTY and has received more than current balance
RECENTLY_ACTIVE	BOOLEAN	Last activity is recent (<i>tunable</i>)
ZOMBIE	BOOLEAN	Was empty and dormant for a long time, then got used again
SCAMMER	BOOLEAN	The owner is marked as a scammer
DISPOSABLE	BOOLEAN	OLD, a few transactions in a short period of time (<i>tunable</i>)
MINER	BOOLEAN	Related to mining activities
BITCOINTALK_USER	STRING	The BitcoinTalk (forum) username of the owner
BITCOINOTC_USER	STRING	The Bitcoin-OTC (exchange) username of the owner

Appendix A (Continued)

A.2 Labels for clusters

In Table VIII we list the labels for clusters with their description.

Appendix A (Continued)

TABLE VIII: LABELS FOR CLUSTERS

Label	Type	Meaning
cluster_id	NUMBER	The ID of the cluster
first_seen	NUMBER	Timestamp of the first appearance of addresses in the cluster
last_seen	NUMBER	Timestamp of the last appearance of addresses in the cluster
recv	NUMBER	Total amount received by addresses in the cluster
sent	NUMBER	Total amount sent from addresses in the cluster
min_balance	NUMBER	Minimum balance of addresses in the cluster
max_balance	NUMBER	Maximum balance of addresses in the cluster
avg_balance	NUMBER	Average balance of addresses in the cluster
n_tx	NUMBER	Number of transactions in which the addresses in the cluster appear
mining	RATIO	Ratio of transactions coming from direct or pooled mining
gambling	RATIO	Ratio of transactions to/from gambling sites
exchanges	RATIO	Ratio of transactions to/from exchanges
wallets	RATIO	Ratio of transactions to/from web wallets
bitcointalk	RATIO	Ratio of transactions to/from known BitcoinTalk users
bitcoinotc	RATIO	Ratio of transactions to/from known Bitcoin-OTC users
freebies	RATIO	Ratio of transactions to/from faucets or other freebies
donations	RATIO	Ratio of transactions to/from known donation addresses
OTA	RATIO	Ratio of One-Time-Addresses in the cluster
OLD	RATIO	Ratio of OLD addresses in the cluster
NEW	RATIO	Ratio of NEW addresses in the cluster
EMPTY	RATIO	Ratio of EMPTY addresses in the cluster
EXHAUSTED	RATIO	Ratio of EXHAUSTED addresses in the cluster
RECENTLY_ACTIVE	RATIO	Ratio of RECENTLY_ACTIVE addresses in the cluster
ZOMBIE	RATIO	Ratio of ZOMBIE addresses in the cluster
SCAMMER	RATIO	Ratio of SCAMMER addresses in the cluster
DISPOSABLE	RATIO	Ratio of DISPOSABLE addresses in the cluster
MINER	RATIO	Ratio of MINER addresses in the cluster
BITCOINTALK_USER	STRING	BitcoinTalk usernames of owners of addresses in the cluster
BITCOINOTC_USER	STRING	Bitcoin-OTC usernames of owners of addresses in the cluster
SCAMMER	BOOLEAN	The owner is marked as a scammer

Appendix B

SQL SCHEMAS

B.1 SQL schema of the blockchain

-- *Tables*

```
CREATE TABLE blocks(  
    block_id BIGINT NOT NULL PRIMARY KEY,  
    block_hash TEXT NOT NULL,  
    time BIGINT NOT NULL  
);  
CREATE TABLE tx(  
    tx_id BIGINT NOT NULL PRIMARY KEY,  
    tx_hash TEXT NOT NULL,  
    block_id BIGINT NOT NULL,  
    FOREIGN KEY (block_id) REFERENCES blocks (block_id)  
);  
CREATE TABLE txin(  
    txin_id BIGINT NOT NULL PRIMARY KEY,  
    txout_id BIGINT NOT NULL,  
    tx_id BIGINT NOT NULL,  
    txin_pos INT NOT NULL,  
    FOREIGN KEY (tx_id) REFERENCES tx (tx_id)  
);  
CREATE TABLE txout(  
    txout_id BIGINT NOT NULL PRIMARY KEY,  
    address CHAR(40),  
    txout_value BIGINT NOT NULL,  
    tx_id BIGINT NOT NULL,  
    txout_pos INT NOT NULL,  
    FOREIGN KEY (tx_id) REFERENCES tx (tx_id)  
);  
-- Indexes  
CREATE INDEX x_txin_txid ON txin (tx_id);  
CREATE INDEX x_txin_txout ON txin (txout_id);  
CREATE INDEX x_txout_address ON txout (address);  
CREATE INDEX x_txout_txid ON txout (tx_id);
```


Appendix B (Continued)

B.2 SQL schema of the features DB

-- *Tables*

```
CREATE TABLE addresses(
address TEXT NOT NULL PRIMARY KEY,
first_seen INT, last_seen INT, recv INT, sent INT, balance INT, n_tx INT, mining REAL,
gambling REAL, exchanges REAL, wallets REAL, bitcointalk REAL, bitcoinotc REAL,
freebies REAL, donations REAL, OTA BOOLEAN, OLD BOOLEAN, NEW BOOLEAN, EMPTY BOOLEAN,
EXHAUSTED BOOLEAN, RECENTLY_ACTIVE BOOLEAN, ZOMBIE BOOLEAN, SCAMMER BOOLEAN,
DISPOSABLE BOOLEAN, MINER BOOLEAN, BITCOINTALK_USER TEXT, BITCOINOTC_USER TEXT,
cluster_id INT);
```

```
CREATE TABLE clusters(
cluster_id INT NOT NULL PRIMARY KEY,
first_seen INT, last_seen INT, recv INT, sent INT, n_tx INT, mining REAL,
gambling REAL, exchanges REAL, wallets REAL, bitcointalk REAL, bitcoinotc REAL,
freebies REAL, donations REAL, OTA REAL, OLD REAL, NEW REAL, EMPTY REAL,
EXHAUSTED REAL, RECENTLY_ACTIVE REAL, ZOMBIE REAL, SCAMMER REAL, DISPOSABLE REAL,
MINER REAL, BITCOINTALK_USER TEXT, BITCOINOTC_USER TEXT, min_balance INT,
max_balance INT, avg_balance INT);
```

-- *Indexes*

```
CREATE INDEX x_cluster_id ON addresses (cluster_id);
CREATE INDEX x_last_seen ON addresses (last_seen);
```

B.3 SQL schema of the trades DB

-- *Tables*

```
CREATE TABLE trades(tid integer, currency text, amount real, price real,
date integer);
```

-- *Indexes*

```
CREATE UNIQUE INDEX trades_currency_tid_index on trades(currency,tid);
```

Appendix C

PYTHON CODE FOR THE CLUSTERING ALGORITHM

```
for tx_id in range(min_txid, max_txid_res + 1):
    # Save progress to files
    if tx_id % 1000000 == 0 and not loaded:
        print("TRANSACTION ID: %d" % (tx_id))
        save(users, FILENAME, tx_id)

    loaded = False

    try:
        in_res = db.query(in_query_addr, (tx_id,))
        out_res = db.query(out_query_addr, (tx_id,))
    except Exception as e:
        print(e)
        continue

    # IN - Heuristic 1 - multi-input transactions
    found = None
    for line in in_res:
        address = line[0]
        if address is None:
            continue
        pos = users.get(address)
        if pos is not None:
            users[address] = pos
            found = pos
        break
    else:
        continue

    if found is None:
        max_cluster_id += 1
        found = max_cluster_id

    for address in in_res:
        users[address[0]] = found
```

Appendix C (Continued)

```
# OUT - Heuristic 2 - shadow addresses
# Exploit bitcoin client bug - "change never last output"
# https://bitcointalk.org/index.php?topic=128042.msg1398752#msg1398752
# https://bitcointalk.org/index.php?topic=136289.msg1451700#msg1451700
if len(out_res) == 2:
    address1 = out_res[0][0]
    address2 = out_res[1][0]
    try:
        appeared1_res = db.query(used_so_far_query, (tx_id, address1),
                                fetch_one=True)
        appeared2_res = db.query(used_so_far_query, (tx_id, address2),
                                fetch_one=True)
    except Exception as e:
        die(e)

    if appeared1_res == 0:
        # Address 1 is never used and appeared, likely a shadow address,
        # add to previous group
        # Exploits bitcoin client bug
        users[address1] = found

    if appeared2_res == 0 and appeared1_res == 1:
        # This is deterministic - last address is actually a shadow address
        users[address2] = found

users = save(users, FILENAME, max_txid_res)
```

Appendix D

LINODE HACK: PRE-THEFT ACTIVITY ON THE NETWORK

Appendix D (Continued)

Thu Jan 26 02:39:59 2012	e26194e8bd0807f9bed840101c23ae185acdaddad8	6bd191d8e556c4d8d035ed30e37497f46a562fa8ba9a6086a351c12a01de3345	7511.83125300 +	498.62000000 =	8010.45125300
Thu Jan 26 17:23:32 2012	13b5a898d925d456409b83f40d0e3fdeb084d9d9	101d0ac9105727ef54ab1b6608c7159599d8063240ef73ca7c1516d129383020	8010.45125300 +	498.57000000 =	8509.02125300
Fri Jan 27 11:10:53 2012	ce0133e0c2a374c2a584912a323d6ce9a1d80ca4	74229d8f41538af607c55afad-c974ac411ecb57377f1c366f1fb00b2db006a3	8509.02125300 +	498.39000000 =	9007.41125300
Fri Jan 27 14:48:17 2012	0402c49bd5a769ebf7d9aad63p9b4fba3437f58c	16933ffc71fb489b8a6a4ebe6798ceec9d16fd491eebcdff59a472e557e697c12d	9007.41125300 -	498.47000000 =	8508.94125300
Fri Jan 27 14:48:17 2012	13b5a898d925d456409b83f40d0e3fdeb084d9d9	16933ffc71fb489b8a6a4ebe6798ceec9d16fd491eebcdff59a472e557e697c12d	8508.94125300 -	498.57000000 =	8010.37125300
Fri Jan 27 14:48:17 2012	813ad1e3b87ad5c11037e5d7d11c2d825a3b3fccc	16933ffc71fb489b8a6a4ebe6798ceec9d16fd491eebcdff59a472e557e697c12d	8010.37125300 -	498.58000000 =	7511.79125300
Fri Jan 27 14:48:17 2012	336e021f04e1f4f8c67ceab702303b7bcd0b09251	16933ffc71fb489b8a6a4ebe6798ceec9d16fd491eebcdff59a472e557e697c12d	7511.79125300 -	498.47000000 =	7013.32125300
...					
Wed Mar 21 11:33:13 2012	4c2a67dc98e6e1f0a998ca4a9086a25725e6a653	9f025354a1dc00e422f93c5b2383e2db87575262790ffdae7e24ffcfe59d5d	2557.32247400 -	58.81247400 =	2498.51000000
Wed Mar 21 11:33:13 2012	eb1379c139bc2cd4cbrad4603e4b1a3738c85cd	9f025354a1dc00e422f93c5b2383e2db87575262790ffdae7e24ffcfe59d5d	2498.51000000 -	2000.00000000 =	498.51000000
Wed Mar 21 11:33:13 2012	cdab6bc68dba5c595bd15bc9d4377ccd399dd5ab9	9f025354a1dc00e422f93c5b2383e2db87575262790ffdae7e24ffcfe59d5d	498.51000000 -	498.51000000 =	0.00000000
transactions = 236					
received = 68546.70216016					
spent = 68546.70216016					
balance = 0.0					
done in 11.474 seconds					

CITED LITERATURE

1. Tygar, J. D. and Yee, B.: Cryptography: It's not just for electronic mail anymore. Technical report, DTIC Document, 1993.
2. Medvinsky, G. and Neumann, B. C.: Electronic currency for the Internet. University of Southern California, Information Sciences Institute, 1993.
3. Dai, W.: B-money. Cypherpunks, 1998.
4. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted, 1:2012, 2008.
5. Francis Elliott, G. D.: Chancellor Alistair Darling on brink of second bailout for banks. The Times, Retrieved May 22, 2013, from <http://www.thetimes.co.uk/tto/business/industries/banking/article2160028.ece>.
6. Haynes, J.: PayPal freezes WikiLeaks account. The Guardian, December, 4 2010.
7. FinCEN: Application of FinCEN's Regulations to Persons Administering, Exchanging, or Using Virtual Currencies. U.S. Department of Treasury, March 2013.
8. Wordpress: Pay Another Way: Bitcoin. Wordpress Blog, <http://en.blog.wordpress.com/2012/11/15/pay-another-way-bitcoin/>.
9. 4Chan: 4Chan Pass. Retrieved May 22, 2013, from <https://www.4chan.org/pass>.
10. WikiLeaks: Donate to WikiLeaks. Retrieved May 22, 2013 from <http://shop.wikileaks.org/donate#dbitcoin>.
11. Olanoff, D.: Reddit Starts Accepting Bitcoin for Reddit Gold Purchases Thanks To Partnership With Coinbase. TechCrunch, Retrieved May 22, 2013 from <http://tcn.ch/174kSXw>.
12. Namecheap: Namecheap Now Accepts Bitcoin. <http://community.namecheap.com/blog/2013/03/05/bitcoin/>.
13. Archive, T. I.: How the Internet Archive is having Great Time with Bitcoin. The Internet Archive Blog, <http://blog.archive.org/2013/04/03/how-the-internet-archive-is-having-great-time-with-bitcoin/>.
14. Christin, N.: Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. arXiv, 2012.
15. Krugman, P. R.: Vehicle Currencies And the Structure Of International Exchange. Working Paper Series, 1979.
16. von Mises, L.: Human Action: A Treatise on Economics. Yale University Press, 2009.

17. Murray N. Rothbard, R. P. M.: Man, Economy, and State: Study Guide. Mises Institute, 2006.
18. Farrell, M.: Bitcoin ATMs coming soon. CNN Money, <http://money.cnn.com/2013/04/04/investing/bitcoin-atms/index.html>.
19. Hypponen, M.: Found: Bitcoin Mining Bot That is Controlled Via Twitter. F-Secure Blog, Retrieved May 22, 2013, from <http://www.f-secure.com/weblog/archives/00002207.html>.
20. Androulaki, E., Karame, G. O., Roeschlin, M., Scherer, T., and Capkun, S.: Evaluating User Privacy in Bitcoin. IACR Cryptology ePrint Archive, 2012.
21. Ian Miers, Christina Garman, M. G. A. D. R.: Zerocoin: Anonymous Distributed e-Cash from Bitcoin. IEEE Symposium on Security and Privacy, 2013.
22. Brugere, I.: Anomaly detection in the Bitcoin transaction network. ESP-IGERT, 2012.
23. Reid, F. and Harrigan, M.: An analysis of anonymity in the bitcoin system. Security and Privacy in Social Networks, pages 197–223, 2013.

VITA

Michele Spagnuolo

Education	<p>B.S., Engineering of Computing Systems Politecnico di Milano 2011</p> <p>M.S., Computer Science (<i>current</i>) University of Illinois at Chicago, Chicago, IL 2013</p> <p>Alta Scuola Politecnica diploma (<i>expected</i>) Politecnico di Milano, Politecnico di Torino 2013</p>
Working experience	<p>Spreaker Security auditing, penetration testing, blackbox analysis of front-end and back-end. 2011 to 2013</p> <p>Gild.com Authoring programming challenges (algorithmic and code snippets). 2012</p>
Projects and business	<p>iPhoneSMSExport.com and SMSRecover.com Founder 2009</p> <p>FriendsGraph.me Co-founder 2012</p> <p>Trovatel.net Founder 2006</p>