Universitat Oberta de Catalunya

Universitat Autònoma de Barcelona

Universitat Rovira i Virgili

Universitat de les Illes Balears

Master Program in Security of Information and Communication Technologies

Marc Santamaria Ortega

The Bitcoin Transaction Graph Anonimity

Master's Thesis

Barcelona, June 14, 2013

Supervisor: PhD Student Cristina Pérez-Solà, Universitat Autònoma de Barcelona

Acknowledgements

I would like to express my sincere gratitude to my advisor PhD Student Cristina Pérez-Solà for her continuous support and her ideas.

Also, I would like to thank my parents and my girlfriend for their motivation and patience, specially these last weeks.

Barcelona, June 14, 2013

Marc Santamaria Ortega

Universitat Oberta de Catalunya
Universitat Autònoma de Barcelona
Universitat Rovira i Virgili
Universitat de les Illes Balears
Master Program in Security of Information and Communication Technologies

ABSTRACT OF
MASTER'S THESIS

Author:	Marc Santamaria Ortega		
Title of thesis:	The Bitcoin Transaction Graph - Anonimity		
Date:	June 14, 2013	**Pages:**	10 + 56
Professorship:	Security in Networks and Systems		
Supervisor:	PhD Student Cristina Pérez-Solà (UAB)		

Bitcoin is a decentralized digital currency based on the use of strong cryptography and an open-source, peer-to-peer internet protocol to operate. The original Bitcoin software was developed by Satoshi Nakamoto (a pseudonym) in 2008 [9]. This thesis looks into the definition of the protocol, taking special interest in how transactions work, in order to determine the anonymity of Bitcoin transactions. Previous works exploring anonymity of Bitcoin will be studied. A major goal of this work is to analyse the anonymity of the Bitcoin network to establish the necessary background for defining address aggregation methods and study the correlation of different addresses that belong to a user. The different aggregation methods which will be specified based in the analysis of some suppositions will be also developed and compared to determine the best mechanism to unite entities. An implementation of the different aggregation methods will be shown as well as some reports which offer a daily analysis of the anonymization methods and services used in transactions of the Bitcoin network.

Keywords: Bitcoin, Bitcoin Transaction Graph, BTG, anonimity aggregation, correlation
Language: English

Contents

List of Tables

List of Figures

List of Algorithms

Symbols and Abbreviations

BTG	Bitcoin Transaction Graph
BTC	Bitcoin (usually used when referring to the currency)

Chapter 1

Introduction

1.1 General

Digital currencies have experienced a boom in the last years. Among them Bitcoin, the first cryptocurrency, has stood out more strongly and is the most widespread. Its money supply its estimated nowadays at around $1 billion USD.

A cryptocurrency consists on a digital currency that relies on cryptography, usually on a proof-of-work scheme, in order to create and handle the currency.

Bitcoin is a decentralized digital currency based on the use of strong cryptography and an open-source, peer-to-peer internet protocol to operate. The original Bitcoin software was developed by Satoshi Nakamoto (a pseudonym) in 2008 [9].

Bitcoin does not have a central bank; it only relies on an internet-based peer-to-peer network. Transactions are made through websites and apps called wallets, and digital signatures are used to process and verify them. It is designed to have no inflation and periodically the number of new bitcoins created is decreased by half, till the year 2140 when a total number of 21 million bitcoins will have been achieved and no more bitcoins will be created.

Because Bitcoin addresses can be obtained as necessary, the correlation between transactions is interesting and has been object of several studies [14] [6] [8] [5]. In this project we will examine how do the Bitcoin network and its transactions work in order to later obtain the Bitcoin Transaction Graph and study the correlation of different addresses that belong to a user.

Some of the main goals of this Thesis will be to improve the existing aggregation methods and identification processes, define new ones and analyze the resulting graph from applying these methods.

1.2 Important Concepts and Design Motivations of Bitcoin

The main aspects that define the Bitcoin currency are:

- It is a cryptocurrency, a type of digital currency that relies on cryptography.
- It is decentralized, so it is not necessary to trust and use banks or other authorities.
- It is peer-to-peer, which means that all the users are equally privileged, everyone can make and validate Bitcoin transactions.
- It is designed to have no inflation and periodically the number of new bitcoins created is decreased by half, till the year 2140 when a total number of 21 million bitcoins will have been achieved and no more bitcoins will be created.
- It prevents double-spending because details of every transaction are broadcasted through the network and included in the blockchain, which contains all transactions ever made.
- It is not possible to modify the blockchain, the blocks are chained in a way that modifying one forces to recompute all the subsequent blocks.

- It is pseudo-anonymous. The Bitcoin addresses are not related to usernames, e-mails or accounts, but every transaction ever made is public so money traceability is possible.

1.3 Why anonymity?

Nowadays society is making use of new technologies to constantly share thoughts, actions, hobbies, ... Most of the time without realizing the reach it will have, especially in the younger generations.

Till now, one of the aspects that had been kept anonymous with more care was the monetary transactions when purchasing goods or services. That is changing with the constant need of sharing everything, but when using Bitcoins is even more noticeable because all the transactions in Bitcoin history are public, due to the design of the protocol. Which means that if a Bitcoin Address is at some point related to a person, everyone will be able to know all the transactions ever made by that person.

There are several design motivations which supposedly help anonymity of users, like the possibility of using a new address for every transaction, or several decisions that a user can take in order to improve its anonymity chances, like keeping different wallets for different purposes, making use of anonymizing services such as VPNs, proxies, TOR, ... but we will discover why they are not enough and why some other measures are needed.

To be able to maintain our anonymity in Internet has been one of the main goals of a lot of researchers and developers not only because of the right we have to protect our personal identity and our personal traits, but also because it is necessary for creating more freedom of expression.

Big companies which their main business consists in the correlation of personal information surely are very interested in linking transactions with the users that made them, whether in dollars, in euros, in Bitcoins or any known currency. This will allow them to know the shopping habits of users, for which services or objects are they willing to pay, ...

Due to all these reasons we should we worried about the anonymity that the currencies we are using grant, and if their anonymity extent is not far, users must know the limitations and possible solutions or actions in order to be able to at least improve the anonymity possibilities.

1.4 Objectives

The main objective of this thesis consists on studying anonymity in the Bitcoin network. To do so it is important to establish first a solid background to correctly understand how Bitcoin transactions work and the features of the protocol.

Other important objectives consist of:

- Explain the Bitcoin protocol, focusing specially on transactions.
- Define the Bitcoin Transaction Graph representing the entities and transactions.
- Analyze previous works in the subject that will allow a better understanding and a broader approach.
- Make assumptions regarding the aggregation of addresses.
- Transform the different assumptions into aggregation methods that can be developed and tested.
- Compare the results from the different aggregation methods.
- Research how to join external information from Internet with Bitcoin addresses.

1.5 Planning

The initial plan defined consisted on the analysis of aggregation methods using an application called Bitcoin Visualizer [17], which uses the graph Database

Neo4j [11], and improving the methods proposed.

Due to the problems encountered when trying to run the application exposed in Appendix B, it was later decided to drop it and do the anonymity analysis and the definition of aggregation methods regardless of Bitcoin Visualizer.

The initial plan consisted on the following stages:

- Introduction to Bitcoin, workplan definition and study of the protocol.
- Analysis of Bitcoin Visualizer and Neo4j and the aggregation methods implemented.
- Anonymization study and proposal of new or improved aggregation methods, and comparison of the results.
- Conclusions and findings, elaboration of the thesis and virtual presentation.

The final plan was:

- Introduction to Bitcoin, workplan definition and study of the protocol.
- Unsuccessful tests with Bitcoin Visualizer, analysis of anonymity in the Bitcoin network.
- Anonymization study, proposal of aggregation methods and development of the different methods.
- Analysis of aggregation results, comparison between them, conclusions and findings. Elaboration of the thesis and virtual presentation.

The major deviations have been caused by the changes made in the study of anonymity aggregations and the reorganization of the plan to adapt to the new scenario.

1.6 Contribution of the thesis

In this thesis a detailed description of Bitcoin from its specification [9] will be given. Special focus will be given to how the Bitcoin transactions work, the build of a graph of the Bitcoin transactions made and the anonymity of the protocol. Also a study of previous works exploring the anonymity issues of the Bitcoin system will be made.

This will establish the knowledge needed for the rest of the sections that will include an anonymity analysis, a definition of different address aggregation methods, a comparison between aggregations, and a study of anonymization methods used.

In the anonymity analysis the claims and anonymity considerations of the specification will be examined, also some possible flaws will be included to try and determine the importance of it.

The definition of different address aggregation methods will allow to explore and analyze the suppositions made in order to determine how anonymous users are in the Bitcoin system and if they make use of all the possibilities offered.

With the comparison between aggregation methods the relevance of anonymity will be studied and verified, and the PROs and CONs of each method compared in order to establish which offers a better mechanism to unite entities.

Finally, an implementation of different reports which will offer a daily analysis of the anonymization methods (VPNs, proxies, TOR, ...) and the services (eWallet, mixing services, ...) used will be done in order to have a better knowledge of the measures taken by the users in order to improve their anonymity and determine the extent of it and its relevance.

1.7 Outline

The thesis consists of six main chapters.

Enclosed in Chapter 1 an introduction to Bitcoin can be found. Also, some important concepts of the Bitcoin protocol and its design motivations are defined. The focus on anonymity is stated afterwards. Finally the contribution of the thesis is specified.

A full overview of Bitcoin [9] is found in Chapter 2. First an introduction to the protocol is given. Then an explanation of how transactions work. Also an exposition of the Bitcoin Transaction Graph is given. Finally an explanation of anonymity in Bitcoin transactions is added to show the importance it should have.

Moreover an anonymity analysis can be found in Chapter 3 where not only the protocol features but also some statements about its anonymity will be analysed. In this Chapter some previous works that exist will also be analyzed and resumed. Finally a study of anonymity in the Bitcoin Network is exposed.

Enclosed in Chapter 4 can be found the analysis of aggregation of addresses. First some suppositions that will be analyzed are introduced. Then, a study of how the different aggregations reflect on the BTG is provided. Also a comparison of the result of the different aggregation methods proposed is supplied to better understand which ones pose a better option for identifying or grouping Bitcoin addresses. Next a study of how users behave when using services and the impact it can have in their anonymity will be done. Finally an analysis of temporal correlations is included.

Several ideas for where future work could be headed can be found in Chapter 5.

At long last the conclusions are given in Chapter 6.

Chapter 2

Bitcoin Description

2.1 Basics

Bitcoin [9] [19] [20] is a decentralized peer-to-peer cryptocurrency that has spread dramatically in the last years, and nowadays is the digital currency most widely used.

Bitcoin relies on cryptography to generate new coins and ensure the correct source and destination of a transaction. New coins are generated when creating blocks, as will be explained in Section 2.2, and is a computationally expensive operation. It is designed this way to make it harder for malicious users to change the transaction history once it has been validated by enough clients.

As it is a decentralized protocol, it is not necessary to rely on banks or other authorities. This is achieved by the construction of the blockchain, which is public and contains the historic of all transactions ever made. By using this public blockchain, all Bitcoin users can check the validity of a transaction and verify the signatures used, because it is based in public key cryptography signatures.

This fits its peer-to-peer background because all users are able to make and validate Bitcoin transactions. It is expected that each user has an updated version of the blockchain before making a transaction so that they can check the validity of the bitcoins they possess. This prevents double-spending because as details of every transaction are broadcasted through the network and included in the blockchain, every user can check if the bitcoins used in a transaction are valid or not. A transaction is considered confirmed once six blocks are recorded after the transaction has been included in a block.

The blockchain is constructed by chaining the blocks in a way that modifying one forces to recompute all subsequent blocks, so it is not possible to easily modify the blockchain. The easier way for an attacker to modify the blockchain would be to create a new block that is deceitful. This would mean that at some point two blockchains with different ending blocks would exist, but for the fake blockchain to prevail, the attacker would need to have more computing power than the rest of the Bitcoin users together. This would be necessary because the predominant blockchain is the one that is verified by a majority of the Bitcoin network's computing power.

It must also be considered that Bitcoin is not an anonymous protocol, at best it could be considered pseudo-anonymous because being the blockchain public, every transaction can be traceable to its origin, although in principle it cannot be linked to a physical person.

Bitcoin is designed to have no inflation and periodically the number of new bitcoins created is decreased by half, till the year 2140 when a total number of 21 million bitcoins will have been achieved and no more bitcoins will be created. At present, each blocks creates 25 bitcoins, and every 4 years more or less, the number of bitcoins that can be mined in a block reduces by 50%.

2.2 Bitcoin transactions

Transactions are one of the bases of the Bitcoin system because they represent the payments between users and its understanding will be primordial to this project because the transaction flow and its analysis is what will allow conjecturing which addresses belong to the same user.

Transactions consist on a data section digitally signed, these data sets are broadcasted to the network and collected into blocks.

Each transaction references previous transactions and specifies a number of Bitcoins that are sent from one Bitcoin address to another.

The format of a Bitcoin transaction inside a Block can be seen in table 2.1:

Table 2.1: Format of Bitcoin transactions

FIELD	DESCRIPTION	SIZE
Version no	Currently 1	4 bytes
In-counter	Positive integer	1 - 9 bytes
List of inputs	The first input of the first transaction is also called "coinbase" (its content was ignored in earlier versions)	<in-counter>-many inputs
Out-counter	Positive integer	1 - 9 bytes
List of outputs	The outputs of the first transaction spend the mined bitcoins for the block	<out-counter>-many outputs
Lock time	If non-zero and sequence numbers are $<$ 0xFFFFFFFF: block height or timestamp when transaction is final	4 bytes

Currently there are three types of transactions:

- To an IP address
- To a Bitcoin address
- Generation of new Bitcoins

New Bitcoins can only be generated by mining a new Block, which contains the last Bitcoin Transactions that have not been recorded before in any block.

Each block records a reference to the previous block, the recent transactions, and also contains the solution to a difficult mathematical puzzle, which is unique to each block. Mining consists in finding this solution, and as the mathematical problem has no efficient solution but is very easy to check if a solution is valid, is very useful to limit the number of coins that can be generated.

This is known as a proof of work system, in Bitcoin this proof consists of choosing a nonce so that the hash of the block including the nonce contains a certain number of zeros at the beginning. The number of zeros of the hash is adjusted over time, in order to adapt the generation of new bitcoins to the current computing capacity. The desired generation rate is one block every 10 minutes, and the difficulty is calibrated every two weeks.

In order to register the "winner" of the mining process, each block contains a record of the Bitcoin address that will receive the coins, and is the first transaction of a block.

Once a user has generated a block he will broadcast it to the network so all the peers can check its validity and add the information to the block they are trying to compute.

The collection of all blocks is known as Bitcoin chain, and the network is designed so that in case that there is a split of the chain only one branch survives. This is done by giving priority to the longest chain of blocks, meaning the one with a higher difficulty.

The format of a Block is shown in table 2.2:

Table 2.2: Format of Bitcoin blocks

FIELD	DESCRIPTION	SIZE
Magic no	value always 0xD9B4BEF9	4 bytes
Blocksize	number of bytes following up to end of block	4 bytes
Blockheader	consists of 6 items	80 bytes
Transaction	counter counter positive integer	1 - 9 bytes
Transactions	the (non empty) list of transactions	<Transaction counter>-many transactions

The block chain basically consists on a transaction database that is shared between all the users of the Bitcoin network; it contains all the transactions ever made in Bitcoins.

If a split of the Bitcoin chain is produced, the transactions in blocks that remain in the short chain that will later be discarded are added to the pool of pending transactions so that they can be added in a posterior block.

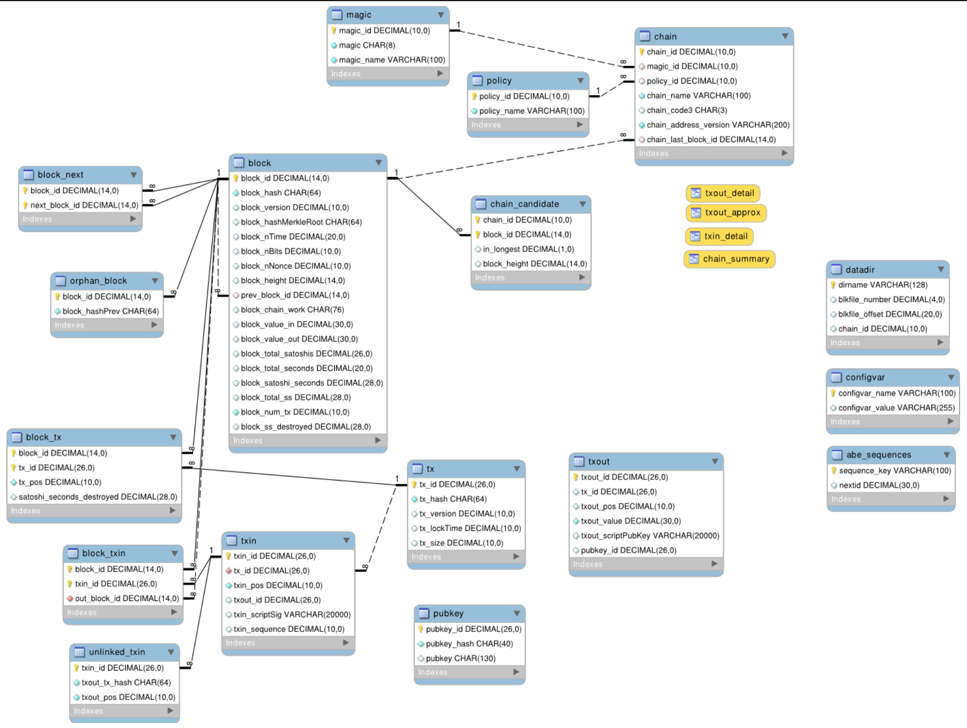
2.3 Bitcoin transaction graph (BTG)

The Bitcoin Transaction Graph (BTG) consists of a Graph representing all the Bitcoin addresses and transactions up to a certain date.

To represent all this information it is important to choose a proper representation method, and also an efficient storage method.

The tutor has provided a MySQL file containing all the information of the Bitcoin chain till the 18th of March 2013.

Figure 2.1 Relational diagram of the Bitcoin Database.



The schema of the database can be found in Figure 2.1.

The analysis will be focused in transactions, so the main tables used will be tx, txin, txout and pubkey. In some analysis the block, block_tx and block_txin will also be used.

The tx table represents all transactions done, its fields are shown in table 2.3.

Table 2.3: Fields of table tx

FIELD	DESCRIPTION	TYPE	KEY
tx_id	Unique id that identifies a transaction	decimal	Primary
tx_hash	Hash of the transaction	char	Unique
tx_version	Version of the transaction	decimal	
tx_lockTime	Indicates when the transaction is final	decimal	
tx_size	Size of the transaction	decimal	

The txin table represents the inputs of all transactions done, its fields are shown in table 2.4, it can be linked with the tx table by the field tx_id.

Table 2.4: Fields of table txin

FIELD	DESCRIPTION	TYPE	KEY
txin_id	Unique id that identifies a transaction input	decimal	Primary
tx_id	Id that identifies a transaction	decimal	Multiple
txin_pos	Position of an input between all the inputs of a transaction	decimal	
txout_id	Id that identifies the output of a transaction	decimal	Multiple
txin_scriptSig	First part of the script - signature	varchar	
txin_sequence	Sequence number for replacement	decimal	

The txout table represents the outputs of all transactions done, its fields are shown in table 2.5, it can be linked with the tx table by the field tx_id and with the txin table by the field txout_id.

The pubkey table represents all the Bitcoin addresses used, its fields are shown in table 2.6, it can be linked with the txout table by the field pubkey_id.

The block tables are block, block_tx and block_txin and represent respectively

Table 2.5: Fields of table txout

FIELD	DESCRIPTION	TYPE	KEY
txout_id	Unique id that identifies a transaction output	decimal	Primary
tx_id	Id that identifies a transaction	decimal	Multiple
txout_pos	Position of an output between all the outputs of a transaction	decimal	
txout_value	Value of BTC	decimal	
txout_scriptPubKey	Second part of the script - pubkey	varchar	
pubkey_id	Id that identifies a Bitcoin address	decimal	Multiple

Table 2.6: Fields of table pubkey

FIELD	DESCRIPTION	TYPE	KEY
pubkey_id	Unique id that identifies a Bitcoin address	decimal	Primary
pubkey_hash	Hash of the pubkey	char	Unique
pubkey	Pubkey	char	

a table representing all the existing blocks, a table that links a block with the transactions included in that block and a table that links a block with the inputs from a transaction.

Chapter 3

Anonimity Analysis

3.1 Introduction

Bitcoin is publicized as an anonymous currency because no bank accounts, email addresses or usernames are necessary to gain or spend Bitcoins, but the truth is that although it allows certain anonymity, the user must be aware of all the implications of his actions in order to maintain his anonymity. Even though the system could support strong anonymity, the current implementation is not very anonymous.

The only identification a certain quantity of Bitcoins has is the Bitcoin address it is associated with, and the public-private key pair that allows signing transactions with this address.

One of the main characteristics to improve anonymity that Bitcoin has been designed with is that it allows each user to generate as many Bitcoin addresses as he wants, allowing even to generate one address for transaction. Another one is that the relation between a user and its public key is stored only in the user's wallet in order to difficult the mapping between users and Bitcoin addresses.

Another way of improving anonymity is by using TOR to connect to the Bitcoin network, but is not fool-prove because in certain situations/attacks transactions can be identified.

But, as has been explained in the definition of the Bitcoin protocol, one of the characteristics of this system is that all the transactions are logged in the Bitcoin chain and are public, so that all the Bitcoins can be traced to its origin, validated and double-spending can be avoided. This means that it is not possible to prevent analysis of the transactions realized till the current date even though at some point changed the system's behaviour, which would be difficult because the system has been designed without a central authority.

In this chapter anonymity features and flaws of the protocol will be examined, first through the analysis of previous works which have explored this area and finally with a study of anonymity in the Bitcoin Network.

3.2 Previous works

First, a review of previous works examining anonymity will be done to establish the basis on which to continue this work.

3.2.1 An Analysis of Anonymity in the Bitcoin System

The first paper analyzing anonymity in the Bitcoin System [14] was first published in July 2011 and later updated in May 2012. In this paper the authors reflect on the construction of two network structures and later they consider the implications of these network structures for anonymity.

In order to build the structures they used all the Bitcoin transactions from the first transaction on the 3rd January 2009 up to the last transaction on the 12th July 2011. This consists of 1019486 transactions between 1253054 unique Bitcoin addresses.

The network structures proposed in the paper are the transaction network and the user network:

- The transaction network represents the flow of Bitcoins between transactions over time. The vertexes of the network represent transactions, and the directed edges represent the outputs and inputs of a transaction, which include the value in Bitcoins and the date.
- The user network represents the flow of Bitcoins between users over time. The vertexes of the network represent users, and the directed edges between an issuer and a receiver represent a transaction, which include the value in Bitcoins and the date.

Later it is explained how both networks are built and some examples of this representation are shown. The transaction network has 974520 vertices and 1558854 directed edges; the transactions that don't have any connection to another transaction are not included in the network because they represent mined Bitcoins or transaction fees not used. As the network does not have multi-edges nor loops, and the output of a transaction will never be an input to the same transaction, it is a directed acyclic graph (DAG).

The user network is first reduced due to the linking that can be assumed when a transaction is done with multi-input, which reveals that all the Bitcoin addresses involved belong to the same user. The initial network has 1253054 vertices (Bitcoin addresses) and 4929950 edges, but the resulting network has 881678 vertices and 1961636 directed edges. Unlike the transaction network, in this case the network has multi-edges, loops and directed cycles.

In the anonymity analysis they explore several ways to deduce information about Bitcoin users.

The first possibility examined consists on integrating off-network information. Since there does not exist a user directory for the Bitcoin system, a partial one could be generated by relating Bitcoin addresses with off-network information. Organization and services that accept Bitcoin transactions, serve as laundry or mixer services, exchange Bitcoins, ... have identifying information of their users (email, credit cards, IP addresses, usernames, shipping addresses, ...) that could be used for dubious purposes. In the paper the use of information provided by the Bitcoin Faucet at the time as well as other sources like public forums, social networks like twitter, ... allows them to link Bitcoin addresses that belong to the same user and at the same point to relate some personal information to some of these addresses.

Next, the analysis of the Bitcoin system from Dan Kaminsky [6] is introduced, which will be studied in the next subsection.

Other anonymity analysis that the authors propose is to make use of network visualization and analysis tools to investigate the flow of Bitcoins and link this information with the data gathered off-network from forums, twitter and webs that donate Bitcoins.

Also they show a case study consisting on the analysis of an alleged theft of 25000 BTC (with a market value at the time of nearly half a million U.S. dollars) on 13/06/2011 at 16:52:23. The theft occurred after the victim's Slush pool account was compromised. In this case they analyze the transactions surrounding the thief to reduce the search space. They discovered that previously the thief had taken 1 BTC probably to do a test before the big blow, and also that there was a fortuitous connection between the victim and the thief trough LulzSec.

One of the more interesting analysis is the flow and temporal one, in which they try to trace significant flows of Bitcoins over time. They take a different approach to the same situation from the previous case study, in which they discover that the flows of Bitcoins after the theft split continuously and later merge, suggesting that the BTC are still controlled by the thief. Also, the involvement of MyBitcoin service in several transactions is shady, as it was previously related to another Bitcoin theft.

Finally they offer other forms of analysis and mitigation strategies; of special interest are the following ideas:

- As many transactions have two outputs which one consist on the return change to the payer knowing the particular client implementation could allow to recognise which is the output and which is the change.
- If several Bitcoin addresses are used at similar times through a certain

period of time they could belong to the same user.

- Bitcoin values converted from other currencies have eight significant digits and could be used to relate Bitcoin addresses and transactions to the exchanges.
- A future version of the Bitcoin protocol should support mixing of Bitcoins to hinder the analysis of user transactions.

3.2.2 Black Ops of TCP/IP 2011

The second work analyzing anonymity that we will study is the analysis of the Bitcoin system from Dan Kaminsky [6] presented at Black Hat 2011.

He analyzes the main flaws of the Bitcoin system in his opinion, the no-scalability of the protocol and its lack of anonymity.

To demonstrate the no-scalability problem he shows that if there were as many transactions with Bitcoins as there are with VISA an average of 1GB per second would be generated for the Block chain, this means that a network node would need a 3TB disk every three weeks, but also it would need around 50 cores to keep up with the flow of information.

If Bitcoins increases to a point that an average computer is not able to be a node it would be necessary to create supernodes, but then you end up shifting the peer-to-peer model to a "regular banking" model.

For the anonymity flaw a different approach as the ones seen on the previous work analyzed is suggested, and also points out that Reid and Harrigan [14] got lucky thanks to how the Bitcoin Faucet worked and a user seeking donations.

The method proposed consists on connecting to every node in the Bitcoin network at once, so that the first node that informs you of a transaction will be the source of that transaction.

The author developed a tool named BlitCoin designed for this purpose.

In order to discover Bitcoin nodes he suggests several approaches:

- Scan the Internet on TCP port 8333
- Join Bitcoin IRC channels
- Recursively ask Bitcoin nodes about all the nodes they know about

Although anonymization services like TOR, proxies, VPNs, ... can be used, in some cases this anonymization can be avoided. For example if you are using TOR but are also listening on port 8333 you could be detected if someone sweeps the Internet, also you could create thousands of nodes to control outbound links

3.2.3 Structure and Anonymity of the Bitcoin Transaction Graph

The third paper studies the structure and anonymity of the Bitcoin transaction graph [8] and was published on May 2013. They analyze dynamical effects and study how some of them increase anonymity while other decrease it.

In their paper the Bitcoin block chain is studied till 6th January 2013, specifically to block number 215,399 and they put focus on an adversary trying to link transactions and discover entities.

The first step is estimating the number of active entities and used public keys (only inputs), and compare them to all the public keys ever used (inputs or outputs) in transactions. They propose the same aggregation method as Reid and Harrigan [14], which consists on linking input addresses. They infer that the increased interest in Bitcoin in several periods of time is related with public trading of bitcoins (April 2010) or the first exchange rate peak (30USD in June/July 2011). It is also deduced from this aggregation method that on average every entity has two public keys.

The relation between public keys or existing keys and entities is also analyzed, and they conclude that due to a greater concern with anonymity the ratio of public keys that can be related to an entity decreases over time.

Their next analysis tries to determine the number of entities and their size over time, stating that entities with several used public keys are infrequent. Also, the distribution of the periods of time in which entities are active is explored, finding that both ends are quite common. A lot of entities are only active during one day, but there are many entities that are active during long periods of time. The best case for maximizing anonymity and avoiding aggregation of addresses would be if entities consisted on a single address and where active for a short period of time.

The authors also investigate the activity of the Bitcoin network, determining how long are entities active over time, and comparing the results with the exchange rate of bitcoins and USD, discovering that the number of active entities is much smaller than the total number of entities and that speculation can be good for anonymity, because the number of active entities increased greatly when the exchange rate reached a peak in June/July 2011. This hypothesis is reinforced with the transaction analysis done in which a peak of transactions is found when the exchange rate peak took place.

It is also indicated that when linking transactions one must be careful with laundry or mixing services which difficult the traceability of bitcoins. And considering the merging method of aggregating bitcoin addresses which are the input of the same transaction they deduce that the periods where this merging is more noticeable is when users where exchanging bitcoins for USD in June/July 2011 and also due to the emergence of the SatoshiDice game.

Finally they study how dormant coins might be harmful for anonymity, because it reduces the anonymity set and is easier to identify entities. It is estimated that nearly a 60% of the bitcoins are dormant, which equals around 6.3 million bitcoins as of January 2013.

The main contributions of the paper are that entity merging is a challenge to Bitcoin anonymity, merged entities follow a scale-free distribution, large entities are less frequent so anonymity increases over time and dormant coins might reduce the anonymity set.

3.2.4 Zerocoin: Anonymous Distributed E-Cash from Bitcoin

The fourth paper proposes an extension to the Bitcoin protocol called Zerocoin [5] published on April 2013. This extension's main objective is to add anonymity to Bitcoin transactions.

This system is proposed so that third party laundry or mixing services are not needed, because these services can be a danger to user's founds or anonymity if for example the operators decide to steal coins, track transactions or the business closes.

Zerocoin is a distributed e-cash system designed to break the relation between Bitcoin transactions without having to trust third parties. It uses cryptographic techniques and the main difference between other e-cash systems is that it is not centralized and each Bitcoin user can generate their own zerocoins.

It uses provably secure cryptographic techniques to guarantee that Bitcoin transactions cannot be traced. Even if a portion of the Bitcoin network was compromised the no-traceability property would still hold because Zerocoin is built on top of Bitcoin, so the Bitcoin protocol should be compromised for the Zerocoin to be affected.

Zerocoin is defined as a separated currency that will work alongside Bitcoin with the same block chain, and zerocoins are changeable with bitcoins. To purchase zerocoins a user has to make a special transaction which includes a special mint, and once this transaction has been validated and is part of the Bitcoin block chain, the same user can transform the zerocoins into bitcoins.

The first step of the process is known as "Zerocoin Mint" and the second part as "Zerocoin Spend", and the extension is defined so that it is not possible to link a Mint transaction with a Spend transaction. It could be said that redeeming zerocoins gives back a completely different set of bitcoins than the ones used initially. This is achieved by using several cryptographic components, mainly digital commitments, one-way accumulators and zero-knowledge proofs.

The authors of the paper define the protocol as the world's biggest laundry, which can handle millions of users, does not need a trusted party and can't be compromised.

The Mint process consists on encrypting a random serial number with a fast commitment algorithm, obtaining a coin that is connected to the chosen number. This "Mint" has to be broadcasted to the Bitcoin network inside a standard Bitcoin transaction which contains in bitcoins the desired equivalent of zerocoins. The block chain will accept this new transaction and add the "Mint" to the global accumulator, so that it cannot be identified among other zerocoin transactions and the currency will not be available except through a Zerocoin Spend.

The Spend process is a bit trickier. In it the client will use the trapdoor (serial number) obtained when the mint was generated to make a new Bitcoin transaction, also adding a zero-knowledge proof of two statements. First, that the user has previously inserted zerocoins in the block chain, and second, that these particular zerocoins contain the serial number used in the Spend transaction. Everyone is able to verify this proof and check that this particular number had never been Spent before, but it does not reveal other information like which particular transaction was the corresponding Mint one.

The Bitcoins used in a Mint transaction form part of a "deposit" that can only be accessed by Bitcoin users that have previously introduced zerocoins to the Bitcoin block chain, and have not used the serial number from their deposit.

The main problems exposed are regarding the time and space needed in order to verify zerocoin transactions and the difficulty of deploying the protocol. They are working in order to reduce the size and cost of verifying the proof and will post an implementation at the end of June. The problem of deploying the protocol is more complicated because it requires that every Bitcoin client is updated to incorporate these changes. Probably several studies and analysis will be made before there is public support of this extension.

3.3 Anonymity in the Bitcoin Network

One of the common points between the analysis of the previous studies and this work is that there are several weak points that make anonymity difficult. The main one is that all the transactions are logged in the Bitcoin chain, which allows seeing the transaction flow. All the Bitcoins can be traced to its origin, even though it can be an unidentified one.

Also, it is only possible to send Bitcoins from an address that has received them, and although one could choose from which address wants to send Bitcoins to another user, the Bitcoin client doesn't allow this which can mean that at some point several Bitcoin addresses can be related to the same user.

A method that has been proposed to anonymize the balance of a user is creating a eWallet account and transfer through there all the Bitcoins to a new address. The main problem with this method is that you have to trust the eWallet service with your coins, and you don't have any guarantee that you will be able to recover them.

Related with the eWallet method, some proposals have been made to use external mixers which don't store the transaction logs in order to protect the identity and origin of the coins sent, but it has the same disadvantage as the previous one because it is indispensable that the Bitcoin user's trust these services.

Keeping anonymity in Internet has always proved difficult when making use of services that expose user information partially. There have been several studies that show attacks in which they learn whether social links exist between different people or social entities [7] or where users from a certain social network can be re-identified in other social networks [10].

The objective of keeping anonymity is double, in one hand it is desirable that a Bitcoin address is not linked to a person, a mail, a username, IP, ... on the other hand that several Bitcoin addresses cannot be related to each other. If only one part is compromised, even though anonymity is in danger the other part can be prevailed with care, but if both sides have been jeopardized it is best to spend or convert all available Bitcoins and start from zero taking more care in how the Bitcoin network is used from that point.

From all this we can observe that maintaining complete anonymity in the Bitcoin network is complicated and with the current protocol is not possible without depending on third parties (eWallet, external mixers, ...) that would act as virtual banks or money exchange. In using these third parties, their reputation and reliability is essential, and they will never be 100% reliable because at any time they can decide to keep all the Bitcoins that they are managing. Also, these centralized services would be capable of identifying and tracking a considerable amount of transactions and analyze user activity.

In Bitcoin undeclared income cannot exist because all the transactions are registered and everyone has access to them. This has several strong points like people not being able to create their own money, or spending twice the same money, but at the same time it does not allow to transfer money from one address to another without being publicly known. For example, in the "real world" one can get cash from his account and give it to anyone or pay in any shop without anyone knowing except the receiver of that money, and the receiver wouldn't even have to know or have any identification of the origin.

I think a method in which two users could send Bitcoins from one to another without having to publicly announce it, but maintaining the controls for double-spending or creating fake coins, and also fulfilling the integrity and confidentiality of a transaction, at least allowing the source to remain hidden is necessary so that Bitcoin becomes a currency more widely used.

All of these problems are mainly caused because Bitcoins are not a tangible asset, and they cannot be taken outside the "economic circuit" and return them to it later. I think it is important to find a workaround for this if third parties are to be avoided in transactions while wanting to ensure anonymity. In some proposals where they suggest external mixers in which logs would be deleted, a user would never have the reliability that they would do so, and in case that the third party keep the logs, it would have a privileged knowledge of the economy and transactions taking place and could later sell this information to the highest bidder (governments, mafias, black-market, ...).

In short, I think that is necessary to design a protocol that allows making transactions more anonymously, where the flow of them is not easily followed, and where no third parties are needed in order to be a real alternative to "conventional" currency, because if not it will always will coexist with it, and depend on it. Although, as has been explored in subsection 3.2.2, if Bitcoin continues growing it will probably reach a point in which it will not be possible for most users to be able to cope with the flow of information, regarding not only the space needed but also the computation required.

Recently, an extension to the Bitcoin protocol has appeared, which is explored in subsection 3.2.4. Although this extension offers great chances of improving anonymity in the Bitcoin network it poses several main problems.

Mainly the computational and space costs, because in the tests done in the Zerocoin paper [5] if a high amount of the transactions are supplanted by Zerocoin transactions the effort needed to calculate a block increases considerably, and could pose a big problem if the number of Bitcoin transactions grows significantly. It could become a choke point before the scalability problem exposed by Dan Kaminsky [6].

Another important problem is the difficulty of deploying the protocol, because it will require collaboration of all the Bitcoin clients, and before it is widely accepted the extension will be probably examined and tested thoroughly.

Also it must be noted, that although transactions could be anonymous once the Zerocoin extension is built in, this will not avoid that all past transactions will continue being weak against aggregation methods, because this extension will only affect future transactions.

Chapter 4

Aggregation of addresses

4.1 Introduction

In the definition of the Bitcoin protocol anonymity was not one of the key features as has been analyzed in the previous section.

In this Chapter the extent of this problem will be explored. In order to do this, several methods to relate Bitcoin Addresses between themselves or obtain additional information from Internet sources akin to particular addresses will be studied.

The objective is to be able to link the Bitcoin addresses so that even if it is not possible to nominally identify a user, it can be known or estimated which Bitcoin addresses are related, and also, in some situations, to gather additional info of the owner of a Bitcoin address like a username in forums, an IP address, an email, a service name, a web, ...

First, some suppositions will be introduced which will help define the aggregation methods and explain its reasoning. Then the different aggregation methods will be explained and it will be shown what new information can be provided with each one.

A comparison of the different methods will follow, analyzing which method offers better possibilities or which outcomes can be mixed in order to obtain better success.

Next an analysis of anonymity of users will be done, examining how they behave when using some services and what information can be gathered.

Finally some temporal correlations will be studied.

4.2 Suppositions

Before being able to define the aggregation methods that will be analyzed, it is important to examine and understand the reasoning behind each one.

4.2.1 Linking of inputs

In previous studies the linking mentioned by Nakamoto [9] has been explored, which basically consists that if a transaction has several inputs, all these inputs will belong to the same owner.

Here this method will be also analyzed as is the most basic but one that has a 100% chance of being correct. More than a supposition, this case is a fact.

4.2.2 Linking of outputs

In most occasions where the transactions have two outputs or more, one of the outputs will belong to the owner and will consist in the change from the transaction.

To identify which output is the change we can suppose that the one with more decimals in the transaction is the change. This assumption will not always be true, but it is the most approximate because usually the decimals of a transaction are reduced due to the following situations:

- The price/amount corresponds to another currency transaction, which will be a quantity that will be several magnitudes bigger than a Satoshi unit.
- The price/amount corresponds to a transaction between users or payment of services/goods, which will usually consist on a number with few decimals to simplify things.
- Prizes tend to be rounded for simplicity and comfort.
- Analyzing prices in Euros/Dollars and other currencies we can easily see that the lower units are rarely used. For example, except when buying things by weight it is unusual to find prices which are not multiples of 5 cents.

One necessary condition for this supposition to be true is that one of the outputs is the change of a transaction. We cannot be sure of it, but we can assume that in most cases it will be this way because it is a quite common occurrence that the bitcoin value of 1 or more addresses of a user don't exactly correspond exactly with the amount of bitcoins that they want to transfer, especially considering how small a satoshi is, which is the base unit of Bitcoin.

In transactions with more outputs we could also make the same assumptions, but the higher the number of outputs, the higher the possibility of choosing a wrong address as the change.

4.2.3 Linking of IPs

Although IP addresses are not stored in the Bitcoin Blockchain it is possible to obtain the IP address used in a transaction.

The easiest way to do so is to run a Bitcoin client and connect to as many nodes as possible as was explored by Dan Kaminsky in Black Hat 2011 [6]. Being a Peer-to-Peer system, if someone is able to connect to all the nodes he will be able to always identify the first node that notifies a transaction and relate its IP with that particular transaction.

It must also be taken in mind that most IP addresses of users are not static and can change overtime, but if several transactions are made with the same IP in a short span of time, it can be assumed that these transactions have been made by the same user.

One way to avoid linking of IPs is the use of anonymizing services like TOR, proxies, VPNs, ... so it is a good idea to let out of the analysis transactions made from IPs that have been identified as an anonymizing service.

4.2.4 Linking of periodic transactions

Usually there are transactions which are made periodically, whether with Euros, Dollars, Bitcoins, ...

It could be interesting to relate these transactions as they will probably be done from different addresses considering the design of the Bitcoin protocol.

In order to aggregate these addresses we can suppose that transactions made over time with the same destination at similar times can be from the same origin. The certainty of this aggregation is lower than with other suppositions.

4.2.5 Linking of mixing/laundry services

The use of mixing or laundry services difficults anonymity analysis due to the inclusion of third parties in the transaction flow. But it is still possible to make some suppositions that will help aggregate user addresses.

Usually, when using mixing or laundry services, a user will want to transfer Bitcoins from one of his addresses to a different one without anyone analyzing the Bitcoin Blockchain being able to identify this transfer.

In order to be able to relate the origin and destination address it will be necessary first to detect the initial transfer to the mixing service and later the final transfer to the destination address.

The easiest supposition is to relate a transaction with a certain amount of coins from the origin address to a transaction with the same amount of coins to the destination address.

In some occasions this supposition will not be enough, and what will have to be linked is the initial transaction with several transactions to the destination node which consist of the same amount after adding all of them.

In this aggregation method it will be very important to recognise and classify all the addresses belonging to mixers or laundries.

4.3 Study of aggregation methods

The methods proposed in the previous section will be analyzed here, and an explanation of how the aggregation process has been carried out will be given.

4.3.1 Aggregation Method 1 - Linking of inputs

This first method, consisting of relating the Bitcoin addresses that appear as an input in the same transaction, although simple in appearance is quite difficult computationally, taking into account the big number of Bitcoin transactions done in all the Bitcoin history.

In this method the first step has been to determine the number of inputs of a transaction. In order to do this the BBDD explained in section 2.3 has been used. The query 4.1 is quite simple and consists on grouping the transaction inputs by the id of the transaction they are an input to.

```
select tx_id, count(*) from txin group by tx_id order by count(*)
    desc
```

Listing 4.1: Query - Number of inputs of transactions

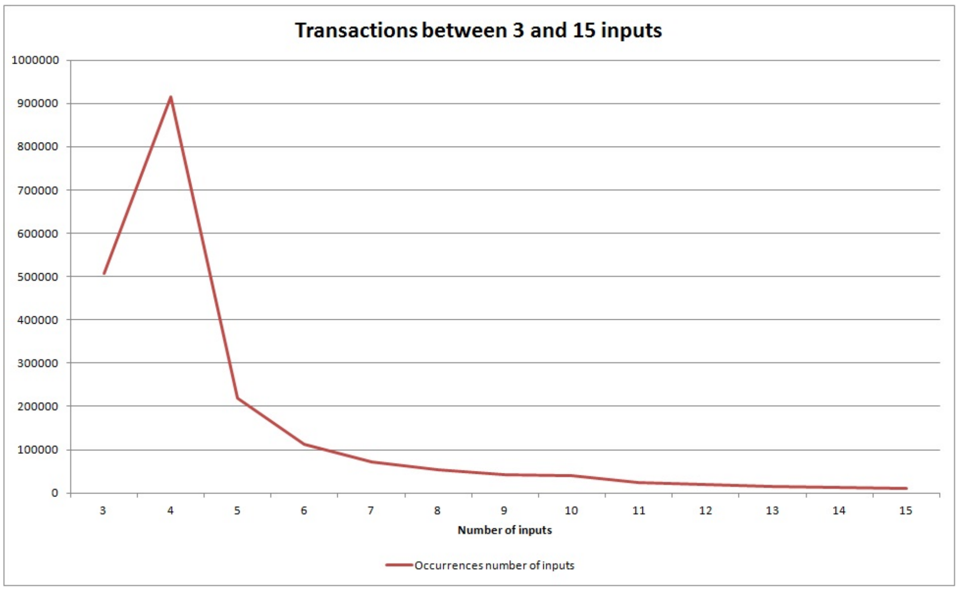
With the list obtained we can observe that nearly a third of all Bitcoin transactions have two or more inputs. From a total of 14515692 different transactions, 4520210 have more than one input, that is a 31,14%.

If we analyze the occurrences of the number of inputs that transactions have, we can easily see that a lot of them have only two inputs (2355366), that represents a 52,1% of the transactions with several inputs, and a 16,22% of all transactions.

With the results from this query we can also see that the total transaction inputs consists of 30086387 elements, and that with only the 21000 transactions that have more inputs (representing a 0,14% of all the transactions) nearly a 10% of all transaction inputs is used (3007435).

In the following charts we will examine the values that are out of the average. The values have been divided in three ranges to be visualized better, which can be found in Figures 4.1 4.2 and 4.3.

Figure 4.1 Ocurrences of transactions who have between 3 and 15 inputs.



It can be observed easily that the transactions with 4, 100, 200 and 250 inputs have a different behaviour. It seems that these transactions could be related with attacks trying to difficult the computing time required for verifying a transaction [3].

In order to accomplish the aggregation method of inputs a script was developed in Bash that aggregates the pubkeys which are inputs of the same transaction. Two versions of this script can be found in Listing A.1 and Listing A.2. The first one stores the results in a file, the other one creates a new table in the

Figure 4.2 Ocurrences of transactions who have between 16 and 80 inputs.

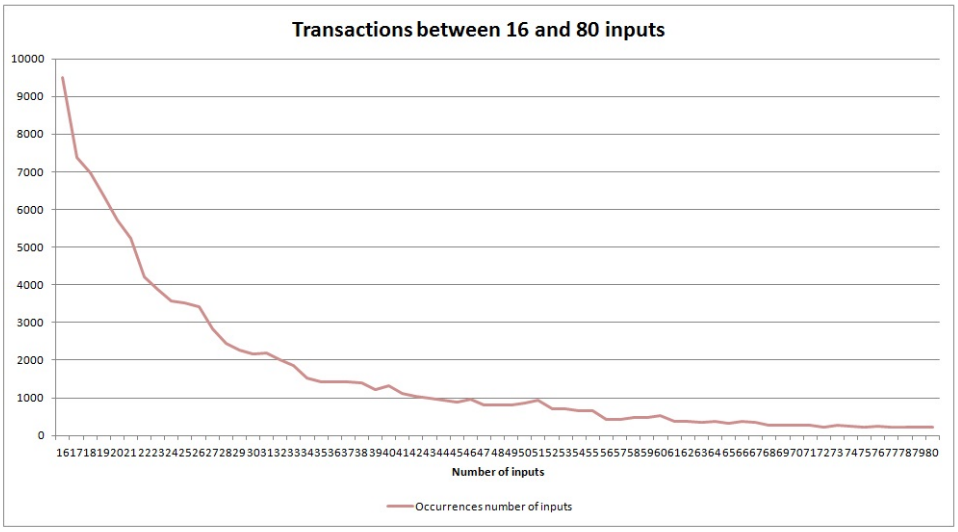
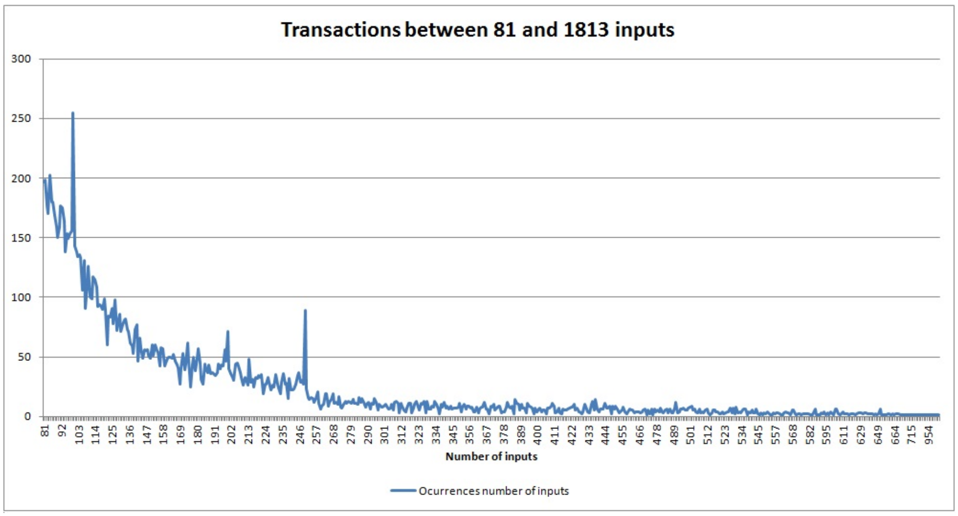


Figure 4.3 Ocurrences of transactions who have between 81 and 1813 inputs.



database and updates its values.

The pseudocode of the script is defined in Algorithm 4.1.

With 5803108 inputs analyzed a total of 1719312 Bitcoin Addresses have been processed and have resulted in only 32956 different identities. It must be taken into account that the transactions analyzed (128240) are the ones with a higher number of inputs, although they only represent the 0,88% of all transactions it corresponds with an analysis of the 19,28% of all the transaction inputs, and 28,88% of all the transaction inputs where a transactions has 2 or more inputs.

Algorithm 4.1 Pseudocode for aggregation method 1 - Linking of inputs

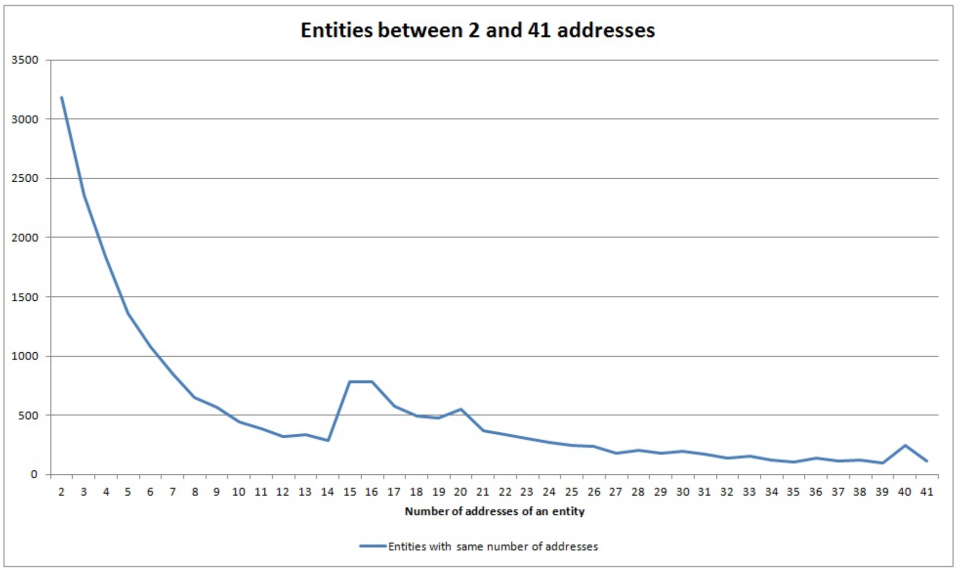
```
$L \Leftarrow$list of all transactions with several inputs
$I \Leftarrow 0$
$PK \Leftarrow$aggregation file where all the pubkeys and their identity will be stored

for all element E of L do
  $PK \Leftarrow$list of pubkeys involved in transaction E
  $F \Leftarrow$pubkeys of PK found on aggregation file
  if $F = 0$ then
    Put PK in aggregation file with identity I
    $I \Leftarrow I + 1$
  else $\{F > 0\}$
    if $F = 1$ then
      $J \Leftarrow$identity of the only pubkey found in the pubkey file
      Put PK not found in aggregation file with identity J
    else $\{F > 1\}$
      $J \Leftarrow$identity of one of the pubkeys found in the pubkey file
      Change identity to J of all pubkeys that have one of the identities
      from the pubkeys found
      Put PK not found in aggregation file with identity J
    end if
  end if
end for
```

The current aggregation has related an average of 52,17 bitcoin addresses per entity. Which is a pretty high number but we have to take into account that the transactions analyzed first are those with a higher number of inputs, that is, the ones resulting in a better aggregation.

It has been detected that in some cases there is no aggregation because in a transaction with several inputs all the inputs are the same bitcoin address. This is usually because it corresponds to small transactions from gambling sites like SatoshiDice, and it can result in a Bitcoin address having a big sum of different inputs of varying value. So far in 6894 cases this situation has arisen.

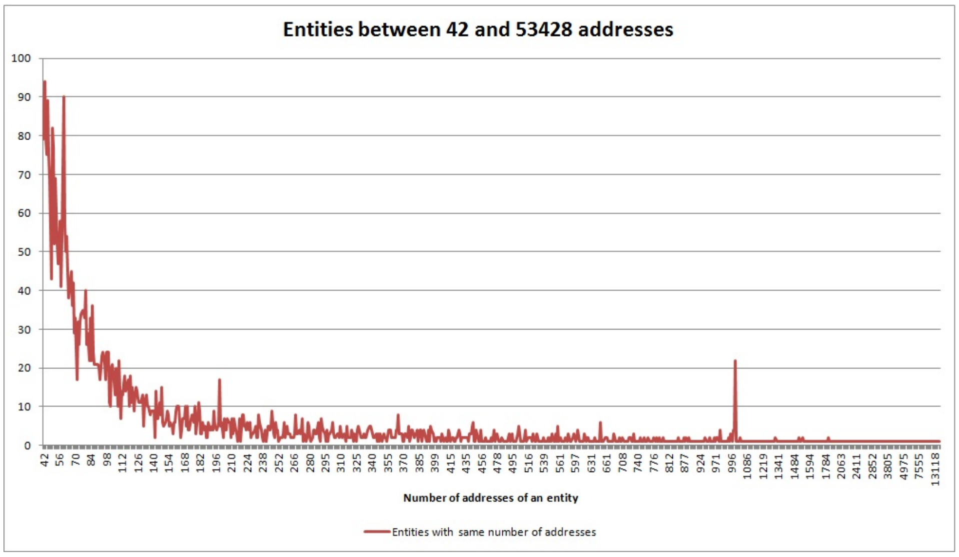
In the following charts we will examine the quantity of entities with the same number of addresses obtained from the script A.1. The values have been divided in two ranges to be visualized better, which can be found in Figures 4.4 and 4.5.

Figure 4.4 Ocurrences of entities who have between 2 and 41 addresses.



In both charts we can see that as expected that entities are more common with less addresses and become less regular the more addresses it has. In Figure 4.4 we can observe that entities between 15 and 20 addresses are more common than other entities similar in size. In Figure 4.5 we can detect that entities with 60, 200 and 1000 addresses are much more common that other near situations, this is probably due to services that have a lot of addresses for its transactions.

Figure 4.5 Ocurrences of entities who have between 42 and 53428 addresses.



4.3.2 Aggregation Method 2 - Linking of outputs

This second method, consisting of relating the Bitcoin addresses that appear as an output with the input of the same transaction, relies on the supposition that in a transaction with more of one output, one of these outputs will correspond with the change of the transaction. The focus is going to be in transactions with only two outputs because the probability of choosing correctly the change is higher.

As in the previous method, this aggregation is quite difficult computationally also, due to the high number of transactions.

In this method the first step has been to determine the number of outputs of a transaction. The query 4.2 is quite simple and consists on grouping the transaction outputs by the id of the transaction they are an input to.

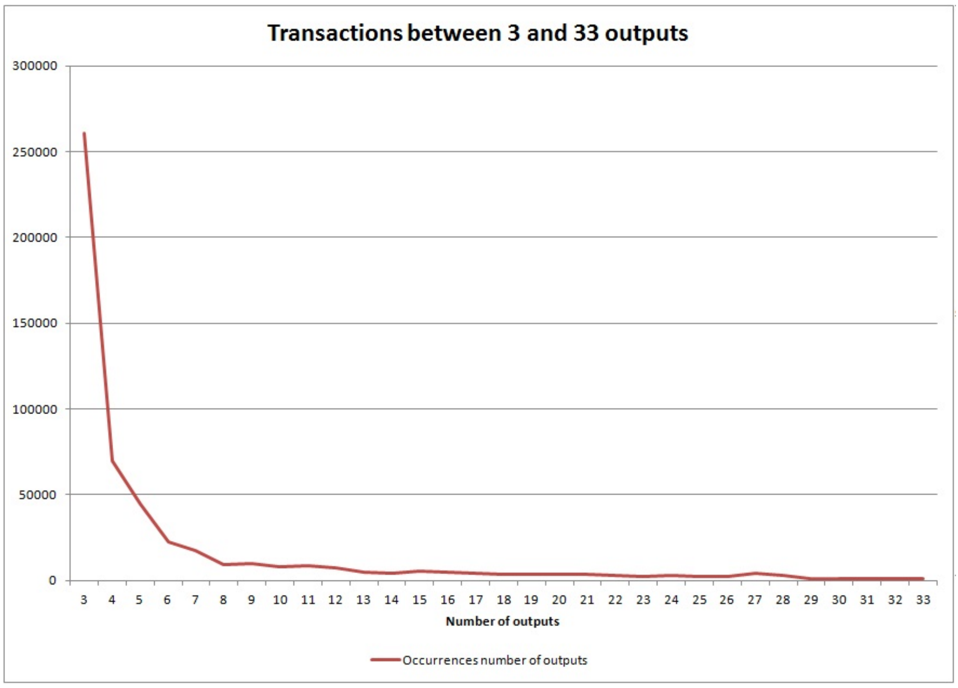
```
select tx_id, count(*) from txout group by tx_id order by count(*)
    desc
```

Listing 4.2: Query - Number of outputs of transactions

With the list obtained we can observe that nearly a 90% of all Bitcoin transactions have two outputs. From a total of 14515692 different transactions, 13061821 have two outputs, that is a 89,98%.

In the following charts we will examine the values that are out of the average. The values have been divided in three ranges to be visualized better, which can be found in Figures 4.6 4.7 and 4.8.

Figure 4.6 Ocurrences of transactions who have between 3 and 15 inputs.



It can be observed easily that the transactions with 40, 41, 51, 65, 81, 101, 102, 201, 257, 301 , 401, 513, 807 and 1803 outputs have a different behaviour. The most interesting case is the one with 41 and 102 outputs, because their occurrences are 8 times higher than near numbers. It would require further analysis to better understand why this particular situation is obtained.

In order to accomplish the aggregation method of outputs a script was developed in Bash similar to the one developed in the previous method that aggregates the pubkeys which are inputs of the same transaction. Two versions of this script can be found in Listing A.3 and Listing A.4. The first one stores the results in a file, the other one creates a new table in the database and updates its values.

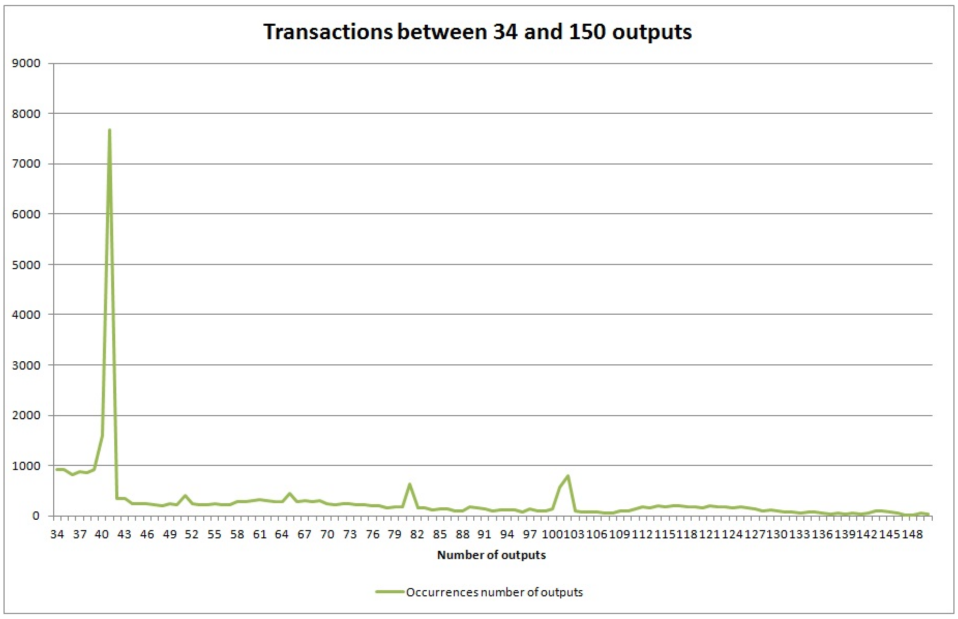
The pseudocode of the script is defined in Algorithm 4.2.

With 274298 transactions analyzed a total of 383990 Bitcoin Addresses have been processed and have resulted in only 32261 different identities. In this

Algorithm 4.2 Pseudocode for aggregation method 2 - Linking of outputs

```
$L$ ⇐list of all transactions with 2 outputs
$I \Leftarrow 0$
$PK$ ⇐aggregation file where all the pubkeys and their identity will be stored

for all element E of L do
    $A$ ⇐find output of E with more decimals
    $B$ ⇐find inputs of E
    $PK$ ⇐list of pubkeys from A and B
    $F$ ⇐pubkeys of PK found on aggregation file
    if $F = 0$ then
        Put PK in aggregation file with identity I
        $I \Leftarrow I + 1$
    else $\{F > 0\}$
        if $F = 1$ then
            $J$ ⇐identity of the only pubkey found in the pubkey file
            Put PK not found in aggregation file with identity J
        else $\{F > 1\}$
            $J$ ⇐identity of one of the pubkeys found in the pubkey file
            Change identity to J of all pubkeys that have one of the identities
            from the pubkeys found
            Put PK not found in aggregation file with identity J
        end if
    end if
end for
```

Figure 4.7 Ocurrences of transactions who have between 16 and 80 inputs.



case the transactions have not been ordered because all of them have the same number of outputs. The 274298 transactions analyzed represent only a 2,09% of all the transactions with two outputs (13061821). Due to the limited resources of the computer used it has not been possible at this moment to advance further the analysis.

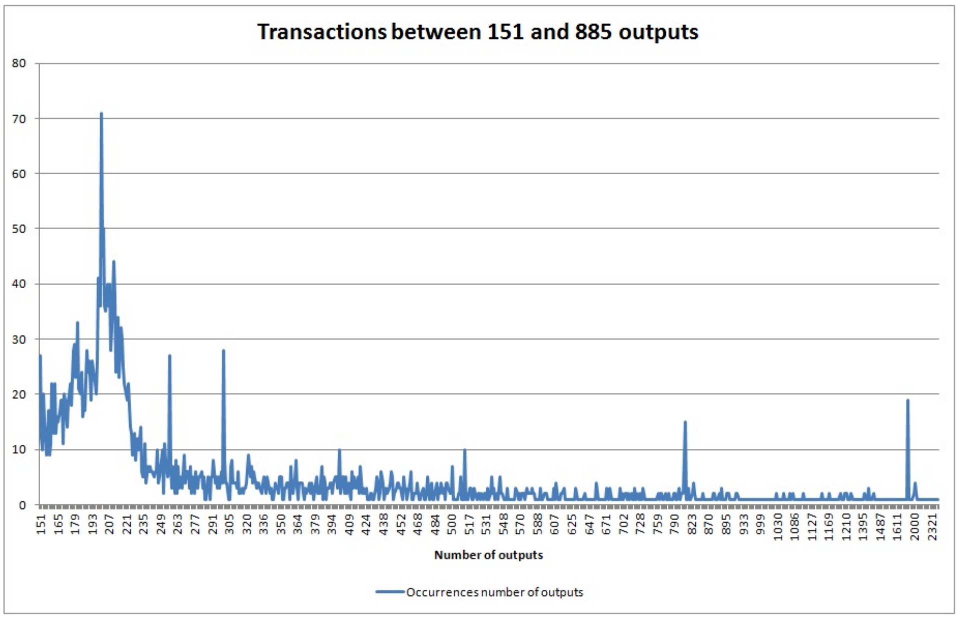
The current aggregation has linked an average of 11,9 bitcoin addresses per entity, we have to consider that in this case, the transactions have not been ordered with any method, so I suppose this aggregation rate will be more or less constant when the analysis is more complete.

It has been detected that in some cases there is no aggregation because in a transaction all the inputs are the same bitcoin address and one of the output address is also the same. This happens in very few cases, so far, only 39 of them have appeared in the transactions analyzed.

In the following charts we will examine the quantity of entities with the same number of addresses obtained from the script A.1. The values have been divided in two ranges to be visualized better, which can be found in Figures 4.4 and 4.5.

In both charts we can see that as expected that entities are more common

Figure 4.8 Ocurrences of transactions who have between 81 and 1813 inputs.



with less addresses and become less regular the more addresses it has. In Figure 4.9 we can observe that entities from 6 addresses and more become very uncommon quickly. In Figure 4.10 we can see that entities continue decreasing in size without big peaks that showed strange situations. The most common case is the one with two addresses per entity with 11808 different entities representing more that a third of all the entities defined (36,6%).

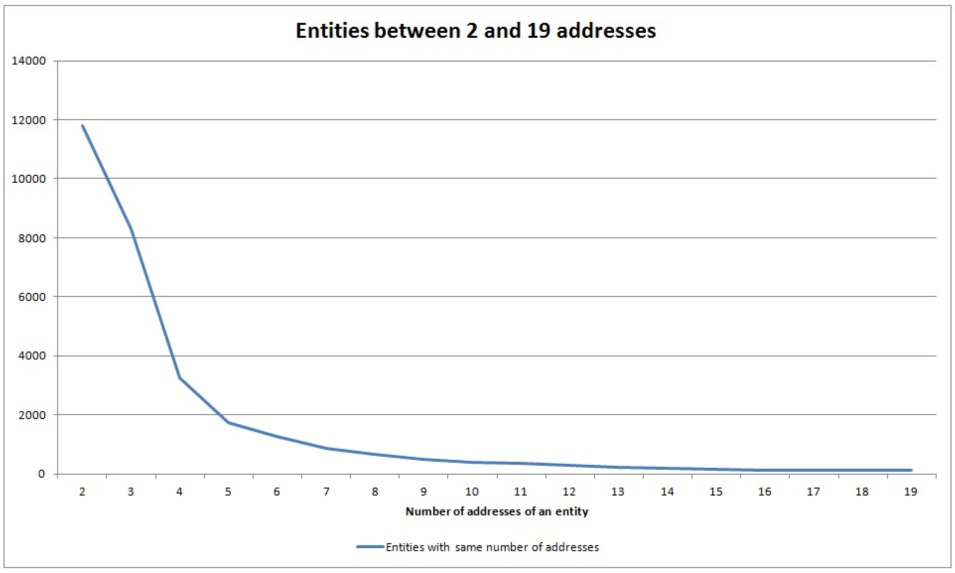
It must be noted that an entity appears with 170472 bitcoin addresses linked. It would be best to wait till the aggregation process has been completed and observe if it continues growing or it has reached its maximum.

4.3.3 Aggregation Method 3 - Linking of IPs

The linking of IPs method depends on the IP information of a transaction. In the BBDD used in other methods the IP address does not appear so it is necessary to use other sources of information.

For this particular analysis we have opted for the web Blockchain [4] where Bitcoin transactions can be explored, and where they offer a lot of services and resources.

Figure 4.9 Ocurrences of entities who have between 2 and 19 addresses.



Several scripts have been developed which download daily all the transactions from the previous day and perform different analysis. These scripts consists of one that downloads all the blocks of the blockchain from yesterday A.6. Another that given a block it descomposes it in transactions A.7. And the last script takes a transaction and stores it in the filesystem so it can be analyzed afterwards A.8.

The first script after it has downloaded all the info from the previous day it computes some statistics. Including total number of transactions, number of anonymizing transactions, percentage, total number of different IPs, number of anonymizing IPs and its percentage.

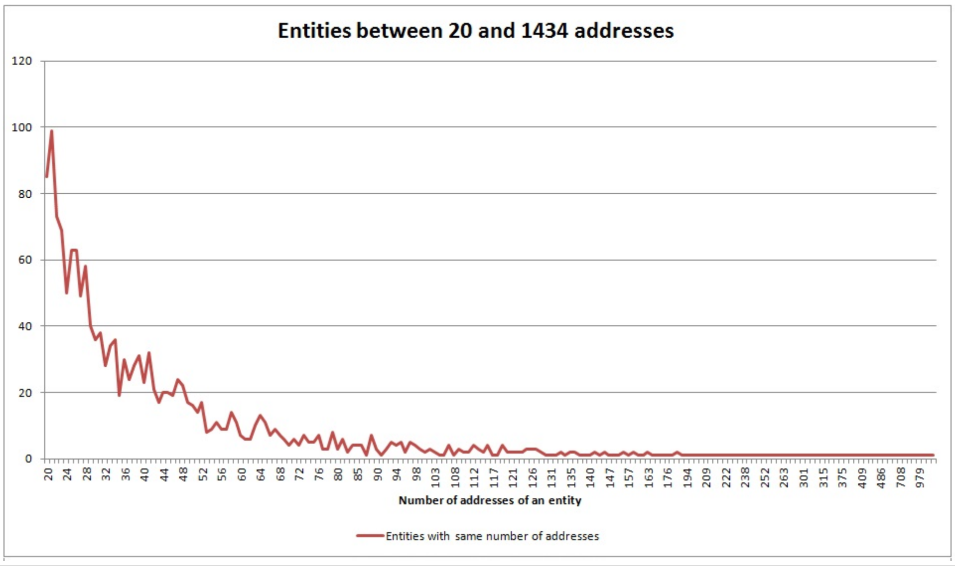
These last few days the number of transactions of a day ranges from 54000 to 69000 and the number of different IPs is between 4100 and 4400.

The IPs most used are 5.9.24.81 and 127.0.0.1, they represent 22,89% (66514) and 18,21% (52921) respectively of all transactions made these last 5 days.

The most used IP belongs to Nogleg [12] as can be discovered with Robtex [15]. The second one represents the Blockchain site [4].

Also to determine which transactions are being made from anonymizing IPs some scripts have been developed to download list from known proxies and

Figure 4.10 Ocurrences of transactions who have between 20 and 1434 inputs.



TOR nodes in order to identify the transactions made from these nodes and treat them differently when analyzing them. The script A.5 consults different sites to obtain free TOR and proxies lists of IPs, it gathers all the information and then creates a file with all of them.

In the last 5 days between 200 and 600 transactions per day have been made with some of the anonymizing IPs identified, they have never surpassed the 1% of all transactions. The IPs used have been around 25-40 different IPs per day.

4.3.4 Aggregation Method 4 - Linking of periodic transactions

With the info collected from the scripts that download the blockchain daily A.6 A.7 A.8 it would be interesting to analyze if some transactions are done periodically taking into account the IPs and/or Bitcoin addresses involved and the amount transferred.

Because the information is downloaded daily it would be easier computationally to calculate the relation of transactions between several days. Also some index could me generated to better navigate through the information.

4.4 Anonymity comparison between aggregation methods

Between the different aggregation methods studied the most reliable is the aggregation of inputs, because is the only one where we are absolutely sure that we are right about the aggregation choosen. In other methods like the aggregation of outputs or the linking of IPs we can never be 100% sure of the suppositions.

That is why when designing the script for computing the aggregation of outputs it was done in a way that it implemented at the same time both the aggregation of inputs and the aggregation of outputs, so in a way, it is a mix of both aggregations. What would be needed after processing the aggregation of outputs, would be to execute the aggregation of inputs over all the transactions where the number of outputs is different from two. This can be computed with the query 4.3, where not only this list of transactions is computed, but also are only chosen orderly the transactions with several inputs.

```
select tx_id, count(*) from txin where tx_id in (select tx_id from
    txout group by tx_id having count(*)!=2) group by tx_id having
    count(*)>1 order by count(*) desc
```

Listing 4.3: Query - Transactions where the number of outputs is not two

From the results obtained so far it seems the aggregation of inputs is more efficient (52,17 vs 11,9), but it must be taken into account that it has an advantage because it begins computing the transactions ordered by the number of inputs. In contrast, the output method where no previous ordering is done it offers a quite good aggregation, but it would be best to dispose of the final computations to make determinant conclusions.

4.5 Study of anonymity of users

In order to study the anonymity of the Bitcoin users it can be helpful to use the scripts developed in subsection 4.3.3. They allow to discover which transactions are being made from IPs belonging to anonymizing services like TOR or proxies. Also, it could be used to obtain the list of addresses and geolocalize them, that way statistics of which countries are most represented in the Bitcoin network, or where is located the biggest flow of Bitcoins would be easy to compute.

To study the anonymity of users in Internet it is useful to monitor services that are common in the Bitcoin community like forums, mixers, laundries, ...

In this project several scripts have been developed to obtain information related with Bitcoin in order to identify users or detect information that can be important.

The first script A.11 consists on a spider that navigates in the last posts from bitcointalk [2] and tries to find Bitcoin addresses and associate them to users. In each post, the first occurrence of an address is asigned to that user, because posterior appareances will probably caused by replies. The particular post an address has been found on is stored so that it can be checked if necessary.

The second script A.12 downloads the latest info posted in Bitbin [1] and tries to find Bitcoin addresses or information related with Bitcoin. It stores not only this information but also the link where it was found.

The third script A.13 does the same as the previous one but for Pastebin [13]. In Pastebin due to the high amount of info being posted constantly and that I was downloading all of it I was banned while I began working on the script. In order to avoid that, some commands were included to select randomly a free proxy from the list created in subsection 4.3.3, this way the ban to my IP could be avoided.

This scripts have proven quite useful, especially the Bitcointalk one. In the forums a lot of users post Bitcoin addresses and is quite easy to link them to the first user that posted the address. For example, in the last two weeks around 4000 Bitcoin addresses have been collected from the forums, both from present posts and some old ones and have been linked to a user of that forum, also, they have been assigned to 1825 different users because some of them include not one but several different addresses through their posts.

The results from bitcointalk should be revised to discover addresses that do not belong to users but instead belong to services. This can be done without much difficulty taking into account how usual is for services and applications to publish their Bitcoin address, usually asking for donations or because it is needed for their business.

All the scripts developed have to be executed every 2 minutes in order to constantly download the latest information, for doing this the best option is to use crontab in Linux so that you can automatically obtain all the information constantly.

These scripts can be easily replicated to do the same in other forums or services used by the Bitcoin community, allowing to correlate the information found in all of them and in that way eliminate the false positives found.

4.6 Analysis temporal correlations

This section is mainly centred in the comparison between the number of transactions overtime and the data used in the aggregation methods of inputs and outputs explored in Section 4.3, to help us determine the reliability of each data considering the historic events.

For this analysis it was necessary to group transactions over time, it was decided to make a month aggregation that would easily show the historic flow of transactions. In order to accomplish this aggregation for the three cases mentioned (all transactions, transactions with several inputs, transactions with two outputs) two scripts were developed in Bash.

The first one calculates all the transactions which belong to each group and the second one calculates the data needed for a chart representation. They can be found in Listings A.9 and A.10 respectively.

From the results obtained, three charts have been drawn to analyze the data. The values have been divided in three ranges to be visualized better, which can be found in Figures 4.11 4.12 and 4.13.

Figure 4.11 Evolution of all transactions, transactions with several inputs and transactions with two outputs between 2009-01 and 2011-04.

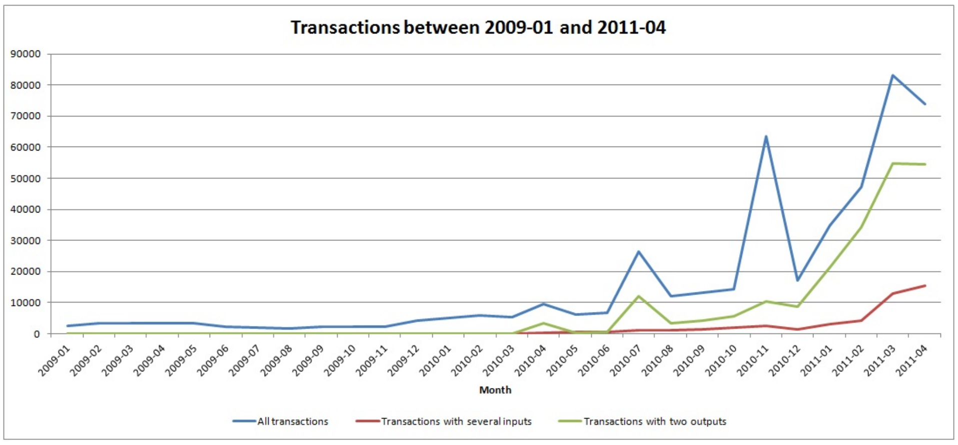
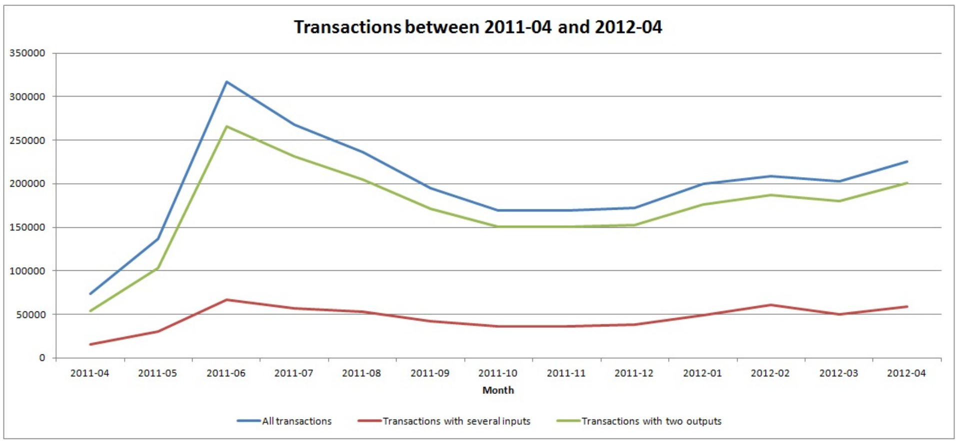
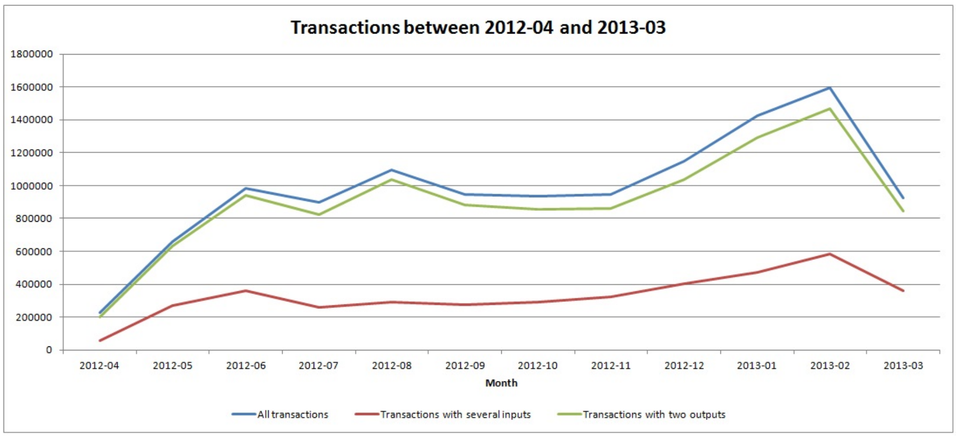


Figure 4.12 Evolution of all transactions, transactions with several inputs and transactions with two outputs between 2011-04 and 2012-04.



In Figure 4.11 several significant peaks can be observed. The first one in July 2010 matches the release of Bitcoin v0.3 which was mentioned on slashdot [18]. The second peak occurred on November 2010 and is related with a single large transaction that was followed with many more trying to hide where the bitcoins were going, and analysis can be found on a paper by D. Ron and A. Shamir [16]. We can assume that the transactions involved moved all the bitcoins from one address to another instead of making a payment and receiving the change in another address due to how the lines representing all the transactions and the transactions with two outputs diverge. The third peak was produced on March 2011 and was related to a low exchange rate BTC/BSD that lasted 6 weeks and which was produced by supposedly automated BTC sales at progressively lower prices. We can also observe in this strange case a split between the flows

Figure 4.13 Evolution of all transactions, transactions with several inputs and transactions with two outputs between 2012-04 and 2013-03.



of transactions.

In Figure 4.12 we can note only one peak, which is related with the exchange rate boom which reached 31.91 USD and also with several thefts that were produced in that month.

In Figure 4.13 we observe one big peak at the beginning of 2013 that corresponds with an increase of the exchange rate of bitcoins which surpassed the previous mark of 31.91 USD, and which continued to rise till the 100 USD reached the 1st of April 2013. We have to take into account that the data studied ends on March 18th 2013, and because of that March has so few transactions. Also, in February over 1 million USD in bitcoins was sold by Coinbase in a single month.

Analyzing the three charts we can conclude that although the transactions with several inputs are very reliable when associating addresses, it is much more useful to focus on transactions with two outputs, because, although there is no 100% certainty that one of the outputs corresponds with the change and it will not be always easy to discern which of the outputs is the change, the amount of transactions is much higher and its evolution has a stronger correlation to the total transactions representation. Regardless of whether there is speculation or not, as can observed in June 2011 and February 2013 where the representation of the evolution of transactions with two outputs is more similar to the evolution of all transactions. In the only cases where more divergence can be found is when there has been notable cases of many automatic transactions which were limited to moving all the bitcoins from one address to another, in one case to hide bitcoins and in the other to force an exchange price drop.

Chapter 5

Future Work

Among the aggregation methods proposed one that has not been fully explored is the linking of mixing/laundry services. Due to lack of time and resources a complete analysis has not been feasible and it would be interesting to follow this path.

In order to better understand how different mixing or laundry services work it would be necessary to do several tests with each of the services to be analyzed. This will require initially to use these services and do several transactions through them, after that it will be very important to make an inventory of as many Bitcoin addresses as possible from these services so that the inner transactions of a mixing or laundry service can be identified and the inputs and outputs of the clients of the services can be easily recognized.

This inventory can be made from the information gathered in the initial tests, with the correlation of addresses done with previous methods and by analyzing the flow of transactions in order to detect patterns.

The objective is to be able to link the origin and destination address of an entity using a mixing or laundry service.

One analysis that remains pending is the union of the complete results from the input and output aggregation methods. Because of how many Bitcoin transactions are and how much time each of the aggregation methods takes, it has not been possible to analyze the entire result of these correlation methods. It would be quite interesting to join both groups of addresses and compute the resulting set so that a better and more complete aggregation can be determined.

After finishing both the aggregation of inputs and outputs from transactions, and their union to get a resulting set, another path to be pursued would be to define a daily process that feeds the database with the new blocks generated and that aggregates the Bitcoin addresses involved in the transactions taking into account the methods defined and the result set obtained from the analysis done till now.

Finally, a visual program could be developed which shows through a timeline the aggregation of entities and the transactions between them while being able to select a certain period of the blockchain, and the detail of the entities displayed. This would allow to analyze correlations from the beginning of the Bitcoin currency to different times in Bitcoin's lifetime and it would be easier to analyze the impact of important historic events for the currency.

Some features could be implemented to show the aggregations that could be done with the knowledge from that period regarding transactions and correlations, and also with all the data from the present moment, so that the behaviour of users can be better analyzed.

Chapter 6

Summary and Conclusions

The main goal of this thesis was to study the anonymity in the Bitcoin network. Initially by understanding how the Bitcoin protocol works, specially the transactions, later by doing an anonymity analysis with a background of previous studies and finally by defining aggregation methods and analyzing its results.

The anonymity analysis has shown that Bitcoin although very useful and secure in preventing attacks or double-spending, has a problem with anonymity that if not resolved could cause that the protocol loses importance if other alternatives that offer more safety in the anonymity field appear.

In Chapter 4 the aggregation methods studied have shown how important it is for users to be careful with the transactions they make if they want to maintain anonymity in the network. There are several options available for this, ranging from being more careful while using the Bitcoin protocol (using TOR to anonymize the IP, keep Bitcoin addresses separated, not abuse of transactions with several inputs, make transactions with more than two outputs,..), making use of mixers or laundry services, ... or even the announced proposed extension Zerocoin [5], which could become an important element in improving anonymity in Bitcoin if it gets enough support.

Among the aggregation methods explored it has been shown that using the relation of several inputs of a transaction, or the link between a payment and its change can be quite useful to generate entities that represent several Bitcoin addresses. The full extent of this analysis has not been reached due to the slowness of the process but it is quite promising.

To difficult both the aggregation of inputs or outputs it depends on the Bitcoin users and how careful they are. To avoid the linking of transactions through the inputs the best method would be to keep different wallets depending on how they will be used, so that addresses from different wallets can never be linked. For preventing the linking of outputs the best option would be to include in each transaction some output with a small value but with a lot of decimals so it difficults the analysis and prevents from easily relating inputs and outputs of a transaction.

It is also important that users are more careful of the information they post in the network, this applies not only to Bitcoin, but lately it seems that a lot of information can be easily obtained from people. In a way, important Bitcoin addresses should be protected as much as a bank account, nobody usually publishes his bank account on the internet, and the same should be done of the main Bitcoin addresses we have, or any address that can be linked to our main ones.

We have to also consider that if the Bitcoin protocol continues to grow at this rate, it will reach a point where some changes will be needed so that it does not die due to its success. As explored by Dan Kaminsky [6] the great quantity of bandwidth, computation and storage required at some point can be detrimental.

The storage problem could be temporally resolved modifying the client so instead of storing the whole blockchain, it only stores the present day and the day before. Every day, the blocks from the day before yesterday would be deleted provided there are enough confirmations, and the first block from yesterday would be accepted as the genesis block. Nowadays there are between 45000 and 69000 transactions daily and every large increase in users occurring will take us closer to that limit. Earlier if some extensions suggested like Zerocoin [5] are implemented, that although very necessary in order to guarantee anonymity, the higher effort needed to compute transactions and validations could produce an earlier choke point.

Bitcoin is a very interesting currency that has grown a lot in the last few years but it still has several challenges to overcome until it can be a real alternative to other legal currency as the dollar or the euro.

References

[1] Bitbin [2013]. *Bitbin.* [Online; accessed 14-June-2013].
URL: *http://www.bitbin.it/index.php*

[2] Bitcointalk [2010]. *Bitcointalk.* [Online; accessed 14-June-2013].
URL: *https://bitcointalk.org/*

[3] Bitcointalk, S. [2013]. *New Bitcoin vulnerability: A transaction that takes at least 3 minutes to verify.* [Online; accessed 14-June-2013].
URL: *https://bitcointalk.org/?topic=140078*

[4] Blockchain [2007]. *Bitcoin Block Explorer.* [Online; accessed 14-June-2013].
URL: *https://blockchain.info/*

[5] I. Miers, C. Garman, M.G. and Rubin, A.D. [2013]. *Zerocoin: Anonymous Distributed E-Cash from Bitcoin. IEEE Symposium on Security and Privacy (Oakland).*

[6] Kaminsky, D. [2011]. *Black Ops of TCP/IP Presentation.*
URL: *http://dankaminsky.com/2011/08/05/bo2k11/*

[7] L. Backstrom, C.D. and Kleinberg, J. [2007]. *Wherefore Art Thou R3579X? Anonymized Social Networks, Hidden Patterns, and Structural Steganography. In Proceedings of the 16th International Conference on World Wide Web*, pages 181–190.

[8] M. Ober, S.K. and Hamacher, K. [2013]. *Structure and Anonymity of the Bitcoin Transaction Graph. Future Internet*, 5:237–250.
URL: *www.mdpi.com/1999-5903/5/2/237/pdf?*

[9] Nakamoto, S. [2009]. *Bitcoin: A Peer-to-Peer Electronic Cash System.*
URL: *http://bitcoin.org/bitcoin.pdf*

[10] Narayanan, A. and Shmatikov, V. [2009]. *De-anonymizing Social Networks. In Proceedings of the 30th Symposium on Security and Privacy,* pages 173–187.

[11] Neo4j [2007]. *Neo4j The Graph Database.* [Online; accessed 14-June-2013].
URL: *http://www.neo4j.org/*

[12] Nogleg [2011]. *Nogleg with Russell Hantz.* [Online; accessed 14-June-2013].
URL: *http://nogleg.com/*

[13] Pastebin [2002]. *Pastebin.* [Online; accessed 14-June-2013].
URL: *http://pastebin.com/*

[14] Reid, F. and Harrigan, M. [2012]. *An Analysis of Anonymity in the Bitcoin System.* doi:arXiv:1107.4524v2[physics.soc-ph].
URL: *http://arxiv.org/pdf/1107.4524v2.pdf*

[15] Robtex [1996]. *Robtex Swiss Army Knife Internet Tool.* [Online; accessed 14-June-2013].
URL: *http://www.robtex.com/*

[16] Ron, D. and Shamir, A. [2012]. *Quantitative Analysis of the Full Bitcoin Transaction Graph. IACR Cryptology.*
URL: *http://eprint.iacr.org/2012/584.pdf*

[17] Russell, J. [2012]. *Bitcoin Visualizer.* [Online; accessed 14-June-2013].
URL: *https://github.com/thallium205/BitcoinVisualizer*

[18] Slashdot [2010]. *Bitcoin-Releases-Version-03.* [Online; accessed 14-June-2013].
URL: *http://news.slashdot.org/story/10/07/11/1747245/Bitcoin-Releases-Version-03*

[19] wiki, B. [2009]. *Bitcoin wiki.* [Online; accessed 14-June-2013].
URL: *https://en.bitcoin.it*

[20] Wikipedia [2009]. *Bitcoin Wikipedia, The Free Encyclopedia.* [Online; accessed 14-June-2013].
URL: *http://en.wikipedia.org/wiki/Bitcoin*

Appendix A

Implementations

In this Appendix can be found all the bash implementations done for the different aggregation methods defined in Section 4.3.

A.1 Aggregation Method 1

For the first aggregation method two scripts have been developed. One stores the aggregation of addresses in a file and the other creates a new table in the database and inserts the aggregations when they are computed.

The only parameters needed in both scripts are the user and password of the database.

```
#!/bin/bash

###Usage: script_aggregation_1_file.sh "user" "password"
###Version: 9
###Aggregation method 1 - Linking of inputs
###It is necessary to include two parameters, the user and
    password of the database.

###Bitcoin addresses not used don't form part of this first
    analysis

###Command to determine % analyzed:
```

```
#date +'%Y/%m/%d %H:%M:%S'; comprobacionTotal="14515692"; echo "
    Total transactions: "$comprobacionTotal;
    comprobacionUltimaTransaction=`tac results_aggregation_1/
    logfile_file.log | grep "tx_id in txin:" | head -n1 | grep -o
    "[0-9]*"`; echo "Last transaction: "
    $comprobacionUltimaTransaction; comprobacionPosicion=`grep -n
    "^$comprobacionUltimaTransaction;" txid_txin_aggregation.txt |
    cut -d":" -f1`; echo "Position in transaction file: "
    $comprobacionPosicion; comprobacionInputsAnalizados=`head -
    n$comprobacionPosicion txid_txin_aggregation.txt | cut -d";" -
    f2 | awk '{SUM+=$1} END {print SUM}'`; echo "Inputs analyzed: "
    $comprobacionInputsAnalizados; comprobacionTotalInputs
    ="30086387"; echo "Total inputs: "$comprobacionTotalInputs;
    comprobacionPorcentage=`echo "scale=2;
    $comprobacionInputsAnalizados*100/$comprobacionTotalInputs" |
    bc -l | sed 's/^\./0\./g'`; echo "Percentage inputs analyzed: "
    $comprobacionPorcentage"%"

###Example:
#2013/06/02 22:18:44
#Total transactions: 14515692
#Last transaction: 12899458
#Position in transaction file: 20154
#Inputs analyzed: 2940362
#Total inputs: 30086387
#Percentage inputs analyzed: 9.77%

###Init data
###In the logFile will be stored the stdout from the script. It
    consists on basic data to determine how much info has been
    analyzed
###In the errorFile will be stored the stderror from the script.
    It allows to detect if there is any problem with the database
    or the program
user=$1
password=$2
BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_aggregation_1
resultFile=`echo $BASEDIR"/results_aggregation_1/
    address_aggregation_1.txt"`
rm $resultFile
touch $resultFile
```

```
logFile=`echo $BASEDIR"/results_aggregation_1/logfile_file.log"`
errorFile=`echo $BASEDIR"/results_aggregation_1/errorfile_file.log
    "`

echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile

###Create transaction file if it does not exist
if [ ! -f $BASEDIR/txid_txin_aggregation.txt ]
then
  mysql -u $user --password="$password" -e "select tx_id, count(*)
     from txin group by tx_id order by count(*) desc" btg | tr '\t'
     ';' > $BASEDIR/txid_txin_aggregation.txt
  grep -v ";1$" $BASEDIR/txid_txin_aggregation.txt > $BASEDIR/
    txid_txin_aggregation_several_inputs.txt
fi

ident=0
cat $BASEDIR/txid_txin_aggregation_several_inputs.txt | grep -v "
    tx_id" | while read line
do
  txid=`echo $line | cut -d";" -f1`
  echo "####################################"
  echo "tx_id in txin: $txid"
  echo "tx_id in txin: "$txid >> $errorFile

  auxiliar1=`mysql -u $user --password="$password" -e "select
    txout_id from txin where tx_id='$txid'" btg | grep -v "txout_id
    " | sort | uniq`
  auxiliar2=`echo "$auxiliar1" | tr '\n' ','`
  auxgrep=`echo "("${auxiliar2:0: -1}")"`
  auxiliarKeys1=`mysql -u $user --password="$password" -e "select
    pubkey_id from txout where txout_id in $auxgrep" btg | grep -v
    "pubkey" | sort | uniq`
  auxiliarKeys2=`echo "$auxiliarKeys1" | tr '\n' '|'`
  auxgrepKeys=`echo "("${auxiliarKeys2:0: -1}")"`

    foundAll=`egrep "^$auxgrepKeys;" $resultFile`
    found=`echo "$foundAll" | cut -d";" -f1`

  if [ -z "$found" ]
  then
```

```
###Case where all the pubkeys involved have not been analyzed before
#The data and a new identifier is added to the aggregation file
     echo "num pubkey found in file EQUAL 0"
     echo "NEW IDENT: $ident"
     auxiliarKeys3=`echo "$auxiliarKeys1" | tr '\n' ','`
     auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
     mysql -u $user --password="$password" -e "select * from pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "pubkey" | tr '\t' ';' | while read bitcoinAddress
     do
       echo $bitcoinAddress";"$ident >> $resultFile
     done
     let ident=$ident+1
   else
         number=`echo "$found" | wc -l`
         if [[ $number -eq 1 ]]
         then
###Case where one pubkeys involved has been analyzed before
       echo "num pubkey found in file EQUAL 1"

#Stablish the value of the analyzed pubkey as reference
               initIdent=`echo "$foundAll" | cut -d";" -f4`
       echo "only IDENT FOUND: $initIdent"
#Add the reference identifier to the pubkeys involved in the current transaction
               auxiliarKeys3=`echo "$auxiliarKeys1" | egrep -v "^$found$" | tr '\n' ','`
               if [ ! -z "$auxiliarKeys3" ]
               then
                    auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
                    mysql -u $user --password="$password" -e "select * from pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "pubkey" | tr '\t' ';' | while read bitcoinAddress
                    do
           echo $bitcoinAddress";"$initIdent >> $resultFile
         done
               fi
     else
###Case where several pubkeys involved have been analyzed before
       echo "num pubkey found in file GREATER THAN 1"

```

```
#Stablish the value of one of these analyzed pubkeys as reference
      initIdent=`echo "$foundAll" | head -n1 | cut -d";" -f4`
      echo "first IDENT FOUND: $initIdent"
#Get list of identifiers belonging to the pubkeys found
            auxChangeValues=`echo "$foundAll" | cut -d";" -f4 |
    sort | uniq`
            auxValues=`echo "$auxChangeValues" | tr '\n' '|'`
            auxGrepValues=`echo "("${auxValues:0: -1}")"`
#Change the identifier of the pubkeys that have the same
    identifier as the pubkeys found to the reference value
            egrep ";$auxGrepValues$" $resultFile | cut -d";" -f1-3
     | sort | uniq | while read line
            do
                echo $line";"$initIdent >> $BASEDIR/
    results_aggregation_1/address_aggregation_1v8.aux
            done
            egrep -v ";$auxGrepValues$" $resultFile >> $BASEDIR/
    results_aggregation_1/address_aggregation_1v8.aux
            mv $BASEDIR/results_aggregation_1/
    address_aggregation_1v8.aux $resultFile
#Add the reference identifier to the pubkeys involved in the
    current transaction
            auxiliarKeys3=`echo "$auxiliarKeys1" | egrep -v "^
    $found$" | tr '\n' ','`
      if [ ! -z "$auxiliarKeys3" ]
            then
              auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
              mysql -u $user --password="$password" -e "select *
    from pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "
    pubkey" | tr '\t' ';' | while read address
                do
                    echo $address";"$initIdent >> $resultFile
                done
      fi
    fi
  fi
echo "######################################"
echo ""
echo ""
echo ""
done >> $logFile 2>> $errorFile
```

```
#How many different entities have been stablished from the Bitcoin
    transactions
cut -d";" -f4 $resultFile | sort | uniq | wc -l

#How many Bitcoin addresses have been used as sources in Bitcoin
    transactions
cut -d";" -f1 $resultFile | sort | uniq | wc -l

#Bitcoin addresses not used don't form part of this first analysis

echo "DATE FI: `date`" > $logFile
echo "DATE FI: `date`" > $errorFile
```

Listing A.1: Script for aggregation method 1 - Linking of inputs - File

```
#!/bin/bash

###Usage: script_aggregation_1_sql.sh "user" "password"
###Version: 9
###Aggregation method 1 - Linking of inputs
###It is necessary to include two parameters, the user and
    password of the database.

###Bitcoin addresses not used don't form part of this first
    analysis

###Command to determine % analyzed:
#date +'%Y/%m/%d %H:%M:%S'; comprobacionTotal="14515692"; echo "
    Total transactions: "$comprobacionTotal;
    comprobacionUltimaTransaction=`tac results_aggregation_1/
    logfile_sql.log | grep "tx_id in txin:" | head -n1 | grep -o
    "[0-9]*"`; echo "Last transaction: "
    $comprobacionUltimaTransaction; comprobacionPosicion=`grep -n
    "^$comprobacionUltimaTransaction;" txid_txin_aggregation.txt |
    cut -d":" -f1`; echo "Position in transaction file: "
    $comprobacionPosicion; comprobacionInputsAnalizados=`head -
    n$comprobacionPosicion txid_txin_aggregation.txt | cut -d";" -
    f2 | awk '{SUM+=$1} END {print SUM}'`; echo "Inputs analyzed: "
    $comprobacionInputsAnalizados; comprobacionTotalInputs
    ="30086387"; echo "Total inputs: "$comprobacionTotalInputs;
    comprobacionPorcentage=`echo "scale=2;
    $comprobacionInputsAnalizados*100/$comprobacionTotalInputs" |
    bc -l | sed 's/^\./0\./g'`; echo "Percentage inputs analyzed: " $comprobacionPorcentage"%"

###Example:
#2013/06/02 22:18:44
#Total transactions: 14515692
#Last transaction: 12899458
#Position in transaction file: 20154
#Inputs analyzed: 2940362
#Total inputs: 30086387
#Percentage inputs analyzed: 9.77%

###Init data
###In the logFile will be stored the stdout from the script. It consists on basic data to determine how much info has been analyzed
###In the errorFile will be stored the stderror from the script. It allows to detect if there is any problem with the database or the program
user=$1
password=$2
BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_aggregation_1
resultFile=`echo $BASEDIR"/results_aggregation_1/address_aggregation_1.txt"`
rm $resultFile
touch $resultFile
logFile=`echo $BASEDIR"/results_aggregation_1/logfile_sql.log"`
errorFile=`echo $BASEDIR"/results_aggregation_1/errorfile_sql.log"`


echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile

###Create aggregation table if it does not exist
#Same fields as pubkey table, but with one more representing the identity
mysql -u $user --password="$password" -e "CREATE TABLE IF NOT EXISTS aggregation1 (
  pubkey_id DECIMAL(26,0) DEFAULT NULL,
  pubkey_hash CHAR(40) NOT NULL,
  pubkey CHAR(130) NULL,
  ident DECIMAL(26,0) NOT NULL,
  PRIMARY KEY (pubkey_id),
  UNIQUE (pubkey_hash)
);" btg

###Create transaction file if it does not exist
if [ ! -f $BASEDIR/txid_txin_aggregation.txt ]
then
  mysql -u $user --password="$password" -e "select tx_id, count(*)
     from txin group by tx_id order by count(*) desc" btg | tr '\t'
     ';' > $BASEDIR/txid_txin_aggregation.txt
  grep -v ";1$" $BASEDIR/txid_txin_aggregation.txt > $BASEDIR/
    txid_txin_aggregation_several_inputs.txt
fi

ident=0
cat $BASEDIR/txid_txin_aggregation_several_inputs.txt | grep -v "
    tx_id" | while read line
do
  txid=`echo $line | cut -d";" -f1`
  echo "####################################"
  echo "tx_id in txin: $txid"
  echo "tx_id in txin: "$txid >> $errorFile

  auxiliar1=`mysql -u $user --password="$password" -e "select
    txout_id from txin where tx_id='$txid'" btg | grep -v "txout_id
    " | sort | uniq`
  auxiliar2=`echo "$auxiliar1" | tr '\n' ','`
  auxgrep=`echo "("${auxiliar2:0: -1}")"`
  auxiliarKeys1=`mysql -u $user --password="$password" -e "select
    pubkey_id from txout where txout_id in $auxgrep" btg | grep -v
    "pubkey" | sort | uniq`
  auxiliarKeys2=`echo "$auxiliarKeys1" | tr '\n' ','`
  auxgrepKeys=`echo "("${auxiliarKeys2:0: -1}")"`

    foundAll=`mysql -u $user --password="$password" -e "select
    pubkey_id,ident from aggregation1 where pubkey_id in
    $auxgrepKeys" btg | tr '\t' ';' | grep -v "pubkey" | sort |
    uniq`
    found=`echo "$foundAll" | cut -d";" -f1`

  if [ -z "$found" ]
  then
###Case where all the pubkeys involved have not been analyzed
    before
#The data and a new identifier is added to the aggregation file
     echo "num pubkey found in file EQUAL 0"
     echo "NEW IDENT: $ident"
     auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" |
     egrep -v "^$" | tr '\n' ','`
     if [ ! -z "$auxiliarKeys3" ]
     then
       auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
       mysql -u $user --password="$password" -e "INSERT INTO
    aggregation1( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
    pubkey_id, pubkey_hash, pubkey, '$ident' AS ident FROM pubkey
    WHERE pubkey_id in $auxsqlKeys" btg
       let ident=$ident+1
     fi
  else
         number=`echo "$found" | wc -l`
         if [[ $number -eq 1 ]]
         then
###Case where one pubkeys involved has been analyzed before
       echo "num pubkey found in file EQUAL 1"

#Stablish the value of the analyzed pubkey as reference
       initIdent=`mysql -u $user --password="$password" -e "select
    ident from aggregation1 where pubkey_id='$found'" btg | grep -v
     "ident"`
       echo "only IDENT FOUND: $initIdent"
#Add the reference identifier to the pubkeys involved in the
    current transaction
       auxiliarKeys3=`echo "$auxiliarKeys1" | egrep -v "^$found$" |
     egrep -v "^$" | tr '\n' ','`
       if [ ! -z "$auxiliarKeys3" ]
       then
         auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
         mysql -u $user --password="$password" -e "INSERT INTO
    aggregation1( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
    pubkey_id, pubkey_hash, pubkey, '$initIdent' AS ident FROM
    pubkey WHERE pubkey_id in $auxsqlKeys" btg
       fi
     else
```

```
###Case where several pubkeys involved have been analyzed before
      echo "num pubkey found in file GREATER THAN 1"

#Stablish the value of one of these analyzed pubkeys as reference
      idents=`echo "$foundAll" | cut -d";" -f2 | sort | uniq`
      initIdent=`echo "$idents" | head -n1`
      echo "first IDENT FOUND: $initIdent"
#Change the identifier of the pubkeys that have the same
   identifier as the pubkeys found to the reference value
      auxChangeValues=`echo "$idents" | egrep -v "^$initIdent$" |
   egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxChangeValues" ]
      then
        auxUpdateValues=`echo "("${auxChangeValues:0: -1}")"`
        mysql -u $user --password="$password" -e "UPDATE
   aggregation1 SET ident='$initIdent' WHERE ident in
   $auxUpdateValues" btg
      fi

#Add the reference identifier to the pubkeys involved in the
   current transaction
      auxiliarKeys4=`echo "$auxiliarKeys1" | egrep -v "^$found$" |
    egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxiliarKeys4" ]
      then
                            auxsqlKeys=`echo "("${auxiliarKeys4:0:
   -1}")"`
        mysql -u $user --password="$password" -e "INSERT INTO
   aggregation1( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
   pubkey_id, pubkey_hash, pubkey, '$initIdent' AS ident FROM
   pubkey WHERE pubkey_id in $auxsqlKeys" btg
      fi
    fi
  fi
echo "###################################"
echo ""
echo ""
echo ""
done >> $logFile 2>> $errorFile


#How many different entities have been stablished from the Bitcoin
```

```
    transactions
cut -d";" -f4 $resultFile | sort | uniq | wc -l

#How many Bitcoin addresses have been used as sources in Bitcoin
    transactions
cut -d";" -f1 $resultFile | sort | uniq | wc -l

#Bitcoin addresses not used don't form part of this first analysis

echo "DATE FI: `date`" > $logFile
echo "DATE FI: `date`" > $errorFile
```

Listing A.2: Script for aggregation method 1 - Linking of inputs - SQL

A.2 Aggregation Method 1

For the second aggregation method two scripts have also been developed, one using files and the other inserting data directly in the database.

The only parameters needed in both scripts are the user and password of the database.

```
#!/bin/bash

###Usage: script_aggregation_2_file.sh "user" "password"
###Version: 3
###Aggregation method 2 - Linking of outputs
###It is necessary to include two parameters, the user and
    password of the database.

###Bitcoin addresses not used don't form part of this first
    analysis

###Command to determine % analyzed:
#date +'%Y/%m/%d %H:%M:%S'; comprobacionTotal="13061821"; echo "
    Total transactions: "$comprobacionTotal;
    comprobacionUltimaTransaccion=`tac results_aggregation_2/
    logfile_file.log | grep "tx_id in txout:" | head -n1 | grep -o
    "[0-9]*"`; echo "Last transaction: "
    $comprobacionUltimaTransaccion; comprobacionPosicion=`egrep -n
    "^$comprobacionUltimaTransaccion$"
    txid_txout_aggregation_double_output.txt | cut -d":" -f1`; echo
     "Position in transaction file: "$comprobacionPosicion;
    comprobacionPorcentage=`echo "scale=2; $comprobacionPosicion
    *100/$comprobacionTotal" | bc -l | sed 's/^\./0\./g'`; echo "
    Percentage transactions analyzed: "$comprobacionPorcentage"%"

###Example:
#2013/06/14 22:18:44
#Total transactions: 13061821
#Last transaction: 1167652
#Position in transaction file: 268016
#Percentage inputs analyzed: 2,05%

###Init data
###In the logFile will be stored the stdout from the script. It
    consists on basic data to determine how much info has been
    analyzed
###In the errorFile will be stored the stderror from the script.
    It allows to detect if there is any problem with the database
    or the program
user=$1
password=$2
BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_aggregation_2
resultFile=`echo $BASEDIR"/results_aggregation_2/
    address_aggregation_2.txt"`
rm $resultFile
touch $resultFile
logFile=`echo $BASEDIR"/results_aggregation_2/logfile_file.log"`
errorFile=`echo $BASEDIR"/results_aggregation_2/errorfile_file.log
    "`

echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile

###Create transaction file if it does not exist
if [ ! -f $BASEDIR/txid_txout_aggregation.txt ]
then
        mysql -u $user --password="$password" -e "select tx_id,
    count(*) from txout group by tx_id order by count(*) desc" btg
    | tr '\t' ';' > $BASEDIR/txid_txout_aggregation.txt
```

```bash
fi

if [ ! -f $BASEDIR/txid_txout_aggregation_double_output.txt ]
then
  cat $BASEDIR/txid_txout_aggregation.txt | grep ";2$" | cut -d";" -f1 > $BASEDIR/txid_txout_aggregation_double_output.txt
fi

ident=0
cat $BASEDIR/txid_txout_aggregation_double_output.txt | grep -v "tx_id" | while read txid
do

  echo "################################"
  echo "tx_id in txout: "$txid
  echo "tx_id in txout: "$txid >> $errorFile

#Obtain pubkey of the outputs
  auxiliarData=`mysql -u $user --password="$password" -e "select pubkey_id,txout_value from txout where tx_id='$txid'" btg | tr '\t' ';' | grep -v "pubkey" | sort | uniq`

#Find the output with more decimals (less 0s)
  auxiliarValue=`echo "$auxiliarData" | cut -d";" -f2 | rev | sort | tail -n1 | rev`
  auxiliarOutput=`echo "$auxiliarData" | egrep -i ";$auxiliarValue$" | head -n1`
  auxiliarKeys0=`echo "$auxiliarOutput" | cut -d";" -f1 | sort | uniq`

#Obtain pubkays of the inputs involved
  auxiliar1=`mysql -u $user --password="$password" -e "select txout_id from txin where tx_id='$txid'" btg | grep -v "txout_id" | sort | uniq`
  auxiliar2=`echo "$auxiliar1" | egrep -v "^$" | tr '\n' ','`
  auxgrep=`echo "("${auxiliar2:0: -1}")"`
  auxiliarKeys1=`mysql -u $user --password="$password" -e "select pubkey_id from txout where txout_id in $auxgrep" btg | grep -v "pubkey" | sort | uniq`

#Check if any of the pubkeys involved has been analyzed before
  auxiliarKeys2=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" |
    egrep -v "^$" | tr '\n' '|'`
  auxgrepKeys=`echo "("${auxiliarKeys2:0: -1}")"`

  foundAll=`egrep "^$auxgrepKeys;" $resultFile`
  found=`echo "$foundAll" | cut -d";" -f1`
  if [ -z "$found" ]
  then
###Case where all the pubkeys involved have not been analyzed
    before
#The data and a new identifier is added to the aggregation file
    echo "inputs or outputs in file EQUAL 0"
    echo "NEW IDENT: $ident"

    auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" |
    egrep -v "^$" | tr '\n' ','`
    if [ ! -z "$auxiliarKeys3" ]
    then
      auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
      mysql -u $user --password="$password" -e "select * from
    pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "pubkey" |
    tr '\t' ';' | while read bitcoinAddress
      do
        echo $bitcoinAddress";"$ident >> $resultFile
      done
      let ident=$ident+1
    fi
  else
    number=`echo "$found" | wc -l`
    if [[ $number -eq 1 ]]
    then
###Case where one pubkeys involved has been analyzed before
      echo "inputs or outputs in file EQUAL 1"

#Stablish the value of the analyzed pubkey as reference
      initIdent=`echo "$foundAll" | cut -d";" -f4`
      echo "only IDENT FOUND: $initIdent"
#Add the reference identifier to the pubkeys involved in the
    current transaction
      auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1"
    | egrep -v "^$found$" | egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxiliarKeys3" ]
      then
```

```bash
          auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
          mysql -u $user --password="$password" -e "select * from pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "pubkey" | tr '\t' ';' | while read bitcoinAddress
          do
            echo $bitcoinAddress";"$initIdent >> $resultFile
          done
        fi
     else
###Case where several pubkeys involved have been analyzed before
       echo "inputs or outputs in file GREATER THAN 1"

#Stablish the value of one of these analyzed pubkeys as reference
            initIdent=`echo "$foundAll" | head -n1 | cut -d";" -f4`

       echo "first IDENT FOUND: $initIdent"
#Get list of identifiers belonging to the pubkeys found
       auxChangeValues=`echo "$foundAll" | cut -d";" -f4 | sort | uniq`
            auxValues=`echo "$auxChangeValues" | egrep -v "^$" | tr '\n' '|'`
            auxGrepValues=`echo "("${auxValues:0: -1}")"`
#Change the identifier of the pubkeys that have the same identifier as the pubkeys found to the reference value
            egrep ";$auxGrepValues$" $resultFile | cut -d";" -f1-3 | sort | uniq | while read line
            do
                echo $line";"$initIdent >> $BASEDIR/results_aggregation_2/address_aggregation_2.aux
            done
            egrep -v ";$auxGrepValues$" $resultFile >> $BASEDIR/results_aggregation_2/address_aggregation_2.aux
            mv $BASEDIR/results_aggregation_2/address_aggregation_2.aux $resultFile
#Add the reference identifier to the pubkeys involved in the current transaction
            auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" | egrep -v "^$found$" | egrep -v "^$" | tr '\n' ','`
       if [ ! -z "$auxiliarKeys3" ]
       then
          auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
```

```
                mysql -u $user --password="$password" -e "select * from pubkey where pubkey_id in $auxsqlKeys" btg | grep -v "pubkey" | tr '\t' ';' | while read address
                do
                    echo $address";"$initIdent >> $resultFile
                done
        fi
      fi
    fi
echo "##################################"
echo ""
echo ""
echo ""
done >> $logFile 2>> $errorFile


#How many different entities have been stablished from the Bitcoin transactions
cut -d";" -f4 $resultFile | sort | uniq | wc -l

#How many Bitcoin addresses have been used as sources in Bitcoin transactions
cut -d";" -f1 $resultFile | sort | uniq | wc -l

echo "DATE FI: `date`" > $logFile
echo "DATE FI: `date`" > $errorFile
```

Listing A.3: Script for aggregation method 2 - Linking of outputs - File

```
#!/bin/bash

###Usage: script_aggregation_2_sql.sh "user" "password"
###Version: 3
###Aggregation method 2 - Linking of outputs
###It is necessary to include two parameters, the user and password of the database.

###Bitcoin addresses not used don't form part of this first analysis

###Command to determine % analyzed:
#date +'%Y/%m/%d %H:%M:%S'; comprobacionTotal="13061821"; echo "Total transactions: "$comprobacionTotal;
```

```
    comprobacionUltimaTransaccion=`tac results_aggregation_2/
    logfile_sql.log | grep "tx_id in txout:" | head -n1 | grep -o
    "[0-9]*"`; echo "Last transaction: "
    $comprobacionUltimaTransaccion; comprobacionPosicion=`egrep -n
    "^$comprobacionUltimaTransaccion$"
    txid_txout_aggregation_double_output.txt | cut -d":" -f1`; echo
     "Position in transaction file: "$comprobacionPosicion;
    comprobacionPorcentage=`echo "scale=2; $comprobacionPosicion
    *100/$comprobacionTotal" | bc -l | sed 's/^\./0\./g'`; echo "
    Percentage transactions analyzed: "$comprobacionPorcentage"%"

###Example:
#2013/06/14 22:18:44
#Total transactions: 13061821
#Last transaction: 1167652
#Position in transaction file: 268016
#Percentage inputs analyzed: 2,05%

###Init data
###In the logFile will be stored the stdout from the script. It
    consists on basic data to determine how much info has been
    analyzed
###In the errorFile will be stored the stderror from the script.
    It allows to detect if there is any problem with the database
    or the program
user=$1
password=$2
BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_aggregation_2
logFile=`echo $BASEDIR"/results_aggregation_2/logfile_sql.log"`
errorFile=`echo $BASEDIR"/results_aggregation_2/errorfile_sql.log"
    `


echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile

###Create aggregation table if it does not exist
#Same fields as pubkey table, but with one more representing the
    identity
mysql -u $user --password="$password" -e "CREATE TABLE IF NOT
    EXISTS aggregation2 (
  pubkey_id DECIMAL(26,0) DEFAULT NULL,
  pubkey_hash CHAR(40) NOT NULL,
  pubkey CHAR(130) NULL,
  ident DECIMAL(26,0) NOT NULL,
  PRIMARY KEY (pubkey_id),
  UNIQUE (pubkey_hash)
);" btg

###Create transaction file if it does not exist
if [ ! -f $BASEDIR/txid_txout_aggregation.txt ]
then
        mysql -u $user --password="$password" -e "select tx_id,
    count(*) from txout group by tx_id order by count(*) desc" btg
    | tr '\t' ';' > $BASEDIR/txid_txout_aggregation.txt
fi

if [ ! -f $BASEDIR/txid_txout_aggregation_double_output.txt ]
then
  cat $BASEDIR/txid_txout_aggregation.txt | grep ";2$" | cut -d";"
    -f1 > $BASEDIR/txid_txout_aggregation_double_output.txt
fi

#Para cada transaccion que tiene unicamente 2 salidas
ident=0
cat $BASEDIR/txid_txout_aggregation_double_output.txt | grep -v "
    tx_id" | while read txid
do

  echo "###################################"
  echo "tx_id in txout: "$txid
  echo "tx_id in txout: "$txid >> $errorFile

#Obtain pubkey of the outputs
  auxiliarData=`mysql -u $user --password="$password" -e "select
    pubkey_id,txout_value from txout where tx_id='$txid'" btg | tr
    '\t' ';' | grep -v "pubkey" | sort | uniq`

#Find the output with more decimals (less 0s)
  auxiliarValue=`echo "$auxiliarData" | cut -d";" -f2 | rev | sort
    | tail -n1 | rev`
  auxiliarOutput=`echo "$auxiliarData" | egrep -i ";
    $auxiliarValue$" | head -n1`
  auxiliarKeys0=`echo "$auxiliarOutput" | cut -d";" -f1 | sort |
    uniq`

#Obtain pubkays of the inputs involved
  auxiliar1=`mysql -u $user --password="$password" -e "select
    txout_id from txin where tx_id='$txid'" btg | grep -v "txout_id
    " | sort | uniq`
  auxiliar2=`echo "$auxiliar1" | egrep -v "^$" | tr '\n' ','`
  auxgrep=`echo "("${auxiliar2:0: -1}")"`
  auxiliarKeys1=`mysql -u $user --password="$password" -e "select
    pubkey_id from txout where txout_id in $auxgrep" btg | grep -v
    "pubkey" | sort | uniq`

#Check if any of the pubkeys involved has been analyzed before
  auxiliarKeys2=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" |
    egrep -v "^$" | tr '\n' ','`
  auxgrepKeys=`echo "("${auxiliarKeys2:0: -1}")"`

  foundAll=`mysql -u $user --password="$password" -e "select
    pubkey_id,ident from aggregation2 where pubkey_id in
    $auxgrepKeys" btg | tr '\t' ';' | grep -v "pubkey" | sort |
    uniq`
  found=`echo "$foundAll" | cut -d";" -f1`
  if [ -z "$found" ]
  then
###Case where all the pubkeys involved have not been analyzed
    before
#The data and a new identifier is added to the aggregation file
     echo "inputs or outputs in file EQUAL 0"
     echo "NEW IDENT: $ident"

     auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1" |
     egrep -v "^$" | tr '\n' ','`
     if [ ! -z "$auxiliarKeys3" ]
     then
       auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
       mysql -u $user --password="$password" -e "INSERT INTO
    aggregation2( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
    pubkey_id, pubkey_hash, pubkey, '$ident' AS ident FROM pubkey
   WHERE pubkey_id in $auxsqlKeys" btg
       let ident=$ident+1
     fi
  else
```

```
    number=`echo "$found" | wc -l`
    if [[ $number -eq 1 ]]
    then
###Case where one pubkeys involved has been analyzed before
      echo "inputs or outputs in file EQUAL 1"

#Stablish the value of the analyzed pubkey as reference
      initIdent=`mysql -u $user --password="$password" -e "select
   ident from aggregation2 where pubkey_id='$found'" btg | grep -v
    "ident"`
      echo "only IDENT FOUND: $initIdent"
#Add the reference identifier to the pubkeys involved in the
   current transaction
      auxiliarKeys3=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1"
    | egrep -v "^$found$" | egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxiliarKeys3" ]
      then
        auxsqlKeys=`echo "("${auxiliarKeys3:0: -1}")"`
        mysql -u $user --password="$password" -e "INSERT INTO
   aggregation2( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
   pubkey_id, pubkey_hash, pubkey, '$initIdent' AS ident FROM
   pubkey WHERE pubkey_id in $auxsqlKeys" btg
      fi
    else
###Case where several pubkeys involved have been analyzed before
      echo "inputs or outputs in file GREATER THAN 1"

#Stablish the value of one of these analyzed pubkeys as reference
            idents=`echo "$foundAll" | cut -d";" -f2 | sort | uniq
   `
      initIdent=`echo "$idents" | head -n1`
      echo "first IDENT FOUND: $initIdent"
#Change the identifier of the pubkeys that have the same
   identifier as the pubkeys found to the reference value
      auxChangeValues=`echo "$idents" | egrep -v "^$initIdent$" |
   egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxChangeValues" ]
      then
        auxUpdateValues=`echo "("${auxChangeValues:0: -1}")"`
        mysql -u $user --password="$password" -e "UPDATE
   aggregation2 SET ident='$initIdent' WHERE ident in
   $auxUpdateValues" btg
```

```
      fi

#Add the reference identifier to the pubkeys involved in the
    current transaction
      auxiliarKeys4=`echo -e "$auxiliarKeys0""\\n""$auxiliarKeys1"
    | egrep -v "^$found$" | egrep -v "^$" | tr '\n' ','`
      if [ ! -z "$auxiliarKeys4" ]
      then
                              auxsqlKeys=`echo "("${auxiliarKeys4:0:
    -1}")"`
        mysql -u $user --password="$password" -e "INSERT INTO
    aggregation2( pubkey_id, pubkey_hash, pubkey, ident ) SELECT
    pubkey_id, pubkey_hash, pubkey, '$initIdent' AS ident FROM
    pubkey WHERE pubkey_id in $auxsqlKeys" btg
      fi
    fi
  fi
echo "#####################################"
echo ""
echo ""
echo ""
done >> $logFile 2>> $errorFile


#How many different entities have been stablished from the Bitcoin
     transactions
cut -d";" -f4 $resultFile | sort | uniq | wc -l

#How many Bitcoin addresses have been used as sources in Bitcoin
    transactions
cut -d";" -f1 $resultFile | sort | uniq | wc -l

echo "DATE FI: `date`" > $logFile
echo "DATE FI: `date`" > $errorFile
```

Listing A.4: Script for aggregation method 2 - Linking of outputs - SQL

A.3 Anonymizing methods

In this script several webs where anonymizing IPs are published have been used. The scripts consults the different sites and downloads the lists. It also joins all of them in a common list with all the anonymization IPs.

```bash
#!/bin/bash

fecha=`date +%Y/%m/%d`

BASEDIR=$(dirname $0)
mkdir -p $BASEDIR/cuaderno/$fecha
mkdir -p $BASEDIR/cuaderno/$fecha/anonimizadores


##########
###TOR###
##########
wget -O /tmp/torlist_dan.txt --referer="http://www.google.com" --user-agent="Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6" --header="Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5" --header="Accept-Language: en-us,en;q=0.5" --header="Accept-Encoding: gzip,deflate" --header="Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7" --header="Keep-Alive: 300" -dnv https://www.dan.me.uk/torlist/

cat /tmp/torlist_dan.txt | sort | uniq | while read line
do
echo $line";dan.me.uk"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt

wget $H='Accept-Language: en-us,en;q=0.5' $H='Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' $H='Connection: keep-alive' -U 'Mozilla/5.0 (Windows NT 5.1; rv:10.0.2) Gecko/20100101 Firefox/10.0.2' --referer=http://www.askapache.com/ "http://proxy.org/tor.shtml" -O /tmp/torlist_proxyorg.txt

html2text /tmp/torlist_proxyorg.txt | egrep "^[0-9]" | tr '\n' '#' | echo -e $(sed 's/&#x/\\x/g') | tr -d ';' | tr '#' '\n' | cut -d" " -f2 | egrep "[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" | while
```

```
    read line
do
echo $line";proxyorg"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt

wget -O /tmp/torlist_resultset_blutmagie.txt http://torstatus.
    blutmagie.de/query_export.php/Tor_query_EXPORT.csv
wget -O /tmp/torlist_ip_blutmagie.txt http://torstatus.blutmagie.
    de/ip_list_all.php/Tor_ip_list_ALL.csv
wget -O /tmp/torlist_exit_blutmagie.txt http://torstatus.blutmagie
    .de/ip_list_exit.php/Tor_ip_list_EXIT.csv

cp /tmp/torlist_resultset_blutmagie.txt $BASEDIR/cuaderno/$fecha/
    anonimizadores/
cp /tmp/torlist_ip_blutmagie.txt $BASEDIR/cuaderno/$fecha/
    anonimizadores/
cp /tmp/torlist_exit_blutmagie.txt $BASEDIR/cuaderno/$fecha/
    anonimizadores/

cat /tmp/torlist_resultset_blutmagie.txt | cut -d"," -f5 | egrep "
    [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*" | sort | uniq | while read line
do
echo $line";blutmagie.resultset"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt

cat /tmp/torlist_ip_blutmagie.txt | cut -d"," -f5 | egrep "
    [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*" | sort | uniq | while read line
do
echo $line";blutmagie.ipall"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt

cat /tmp/torlist_exit_blutmagie.txt | cut -d"," -f5 | egrep "
    [0-9]*\.[0-9]*\.[0-9]*\.[0-9]*" | sort | uniq | while read line
do
echo $line";blutmagie.ipexit"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt

cat $BASEDIR/cuaderno/$fecha/anonimizadores/torlist.txt | cut -d";
    " -f1 | sort | uniq > $BASEDIR/cuaderno/$fecha/anonimizadores/
    toriplist.txt

rm /tmp/torlist*
```

```

##############
###PROXIES###
##############
i=1
num=1
rm /tmp/proxylist_vpngeeks3.txt
while [ $num -ge 1 ]
do
  url_vpngeeks=`echo "http://www.vpngeeks.com/proxylist.php?from="
    $i"&#pagination"`
  wget $url_vpngeeks -O /tmp/proxylist_vpngeeks.txt
  html2text /tmp/proxylist_vpngeeks.txt > /tmp/proxylist_vpngeeks2
    .txt
  num=`cat /tmp/proxylist_vpngeeks2.txt | egrep "
    [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" | wc -l`
  cat /tmp/proxylist_vpngeeks2.txt | egrep "
    [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+"  >> $BASEDIR/cuaderno/$fecha/
    anonimizadores/proxylist_vpngeeks.txt
  cat /tmp/proxylist_vpngeeks2.txt | egrep "
    [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+" | cut -d"|" -f2 | tr -d '_' >>
    /tmp/proxylist_vpngeeks3.txt
  let i=$i+50
  echo "###########"$i;
  echo "###########"$num;
done

cat /tmp/proxylist_vpngeeks3.txt | sort | uniq | while read line
do
echo $line";vpngeeks"
done >> $BASEDIR/cuaderno/$fecha/anonimizadores/proxylist.txt

cat $BASEDIR/cuaderno/$fecha/anonimizadores/proxylist.txt | cut -d
    ";" -f1 | sort | uniq > $BASEDIR/cuaderno/$fecha/anonimizadores
    /proxyiplist.txt


#Master list unifying the TOR, proxies and VPN IP lists
cat $BASEDIR/cuaderno/$fecha/anonimizadores/*iplist.txt | sort |
    uniq | while read line
do
```

```
  auxlist=`grep "^$line$" $BASEDIR/cuaderno/$fecha/anonimizadores
    /*iplist.txt`
  auxlist2=`echo "$auxlist" | cut -d"/" -f13 | cut -d"." -f1 | tr
    '\n' '|'`
  list=`echo ${auxlist2:0: -1}`
  echo $line";"$list
done > $BASEDIR/cuaderno/$fecha/anonimizadores/masterlist.txt

rm /tmp/proxylist_*
```

Listing A.5: Script for obtaining anonymization IPs

A.4 Downloading transactions of the last day

Several scripts have been developed for downloading the transactions from the last day . The first one A.6 downloads all the last blocks of the blockchain till it finds one from the day before yesterday, and for the blocks downloaded that are from yesterday executes the script A.7. This second script takes a block and separates it in transactions which are then sent to script A.8. The last script takes a transaction and stores it in the filesystem so it can be analyzed afterwards.

The first script after it has downloaded all the info from the previous day it computes some statistics.

```
#!/bin/bash

################
###FUNCTIONS###
################

#Recursive function to get the blocks from yesterday
yesterdayBlock() {
  local localHashBlock=$1
  wget -q http://blockchain.info/rawblock/$localHashBlock -O /tmp/
    aux_block_$localHashBlock.txt
  localTimeBlock=`grep "time" /tmp/aux_block_$localHashBlock.txt |
     head -n1 | cut -d":" -f2 | cut -d"," -f1`
  localDateBlock=`date +'%Y-%m-%d' --date="@${localTimeBlock}"`
  localhashPreviousBlock=`grep "prev_block" /tmp/aux_block_$localHashBlock.txt | head -n1 | cut -d'"' -f4`

  if [ "$today" == "$localDateBlock" ]
  then
    yesterdayBlock $localhashPreviousBlock
  elif [ "$yesterday" == "$localDateBlock" ]
  then
    yesterdayBlock $localhashPreviousBlock
    $BASEDIR/script_extract_transactions_from_block.sh "/tmp/aux_block_$localHashBlock.txt"
  fi
  rm /tmp/aux_block_$localHashBlock.txt
  return 0
}

################
###MAIN CODE###
################

#Create the folder where we will save the transactions
yesterday=`date +%Y/%m/%d --date=yesterday`
BASEDIR=$(dirname $0)
mkdir $BASEDIR/cuaderno/$yesterday/transacciones

#Save the date from yesterday
today=`date +'%Y-%m-%d'`
yesterday=`date +'%Y-%m-%d' --date=yesterday`
yesterdaySlash=`date +%Y/%m/%d --date=yesterday`

#Obtain latest block from blockchain.info
wget -q http://blockchain.info/latestblock -O /tmp/aux_latestblock.txt

#Get the hash of the last block
hashLastBlock=`grep "hash" /tmp/aux_latestblock.txt | head -n1 | cut -d'"' -f4`

#Obtain info from the last block
wget -q http://blockchain.info/rawblock/$hashLastBlock -O /tmp/aux_block_$hashLastBlock.txt

```

```
hashPreviousBlock=`grep "prev_block" /tmp/aux_block_$hashLastBlock
    .txt | cut -d'"' -f4`

#Call recursive function
yesterdayBlock $hashPreviousBlock

rm /tmp/aux_latestblock.txt
rm /tmp/aux_block_$hashLastBlock.txt


###Different IPs used
cat $BASEDIR/cuaderno/$yesterdaySlash/transacciones/simple.txt |
    cut -d";" -f3 | sort | uniq -c | sort -nr | awk '{print $2 ";"
    $1}' > $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    ip_transacciones_diferentes.txt


###IPs of anoniymizing services
cat $BASEDIR/cuaderno/$yesterdaySlash/transacciones/simple.txt |
    cut -d";" -f3 | sort | uniq | while read line
do
  list=`grep "^$line;" $BASEDIR/cuaderno/$yesterdaySlash/
    anonimizadores/masterlist.txt`
  result=`echo $?`
  if [ "$result" == "0" ]
  then
    echo $list
  fi
done > $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    ip_transacciones_anonimizadoras.txt


#Anonymizing transactions
cat $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    ip_transacciones_anonimizadoras.txt | cut -d";" -f1 | sort |
    uniq | while read line
do
  num=`grep ";$line;" $BASEDIR/cuaderno/$yesterdaySlash/
    transacciones/simple.txt | wc -l`
  echo $line";"$num
done > $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    transacciones_anonimizadoras.txt
```

```


#Statistics
totalTrans=`wc -l $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    simple.txt | cut -d" " -f1`
echo "total_transactions;"$totalTrans >> $BASEDIR/cuaderno/
    $yesterdaySlash/transacciones/estadisticas_transacciones.txt

numberAnoTrans=`cat $BASEDIR/cuaderno/$yesterdaySlash/
    transacciones/transacciones_anonimizadoras.txt | cut -d";" -f2
    | awk '{SUM=SUM+$1} END {print SUM}'`
echo "number_anonymizing_transactions;"$numberAnoTrans >> $BASEDIR
    /cuaderno/$yesterdaySlash/transacciones/
    estadisticas_transacciones.txt

percentage=`echo "scale=2; $numberAnoTrans*100/$totalTrans" | bc -
    l | sed 's/^\./0\./g'`
echo "percentage_anonymizing_transactions;"$percentage >> $BASEDIR
    /cuaderno/$yesterdaySlash/transacciones/
    estadisticas_transacciones.txt

totalIPs=`wc -l $BASEDIR/cuaderno/$yesterdaySlash/transacciones/
    ip_transacciones_diferentes.txt | cut -d" " -f1`
echo "total_ips;"$totalIPs >> $BASEDIR/cuaderno/$yesterdaySlash/
    transacciones/estadisticas_transacciones.txt

numberAnoIPs=`wc -l $BASEDIR/cuaderno/$yesterdaySlash/
    transacciones/ip_transacciones_anonimizadoras.txt | cut -d" " -
    f1`
echo "number_anonymizing_ips;"$numberAnoIPs >> $BASEDIR/cuaderno/
    $yesterdaySlash/transacciones/estadisticas_transacciones.txt

percentageIPs=`echo "scale=2; $numberAnoIPs*100/$totalIPs" | bc -l
     | sed 's/^\./0\./g'`
echo "percentage_anonymizing_IPs;"$percentageIPs >> $BASEDIR/
    cuaderno/$yesterdaySlash/transacciones/
    estadisticas_transacciones.txt
```

Listing A.6: Script for downloading the blocks from yesterday

```
#!/bin/bash

blockFile=$1
```

```
BASEDIR=$(dirname $0)

cat $blockFile | sed 's/^[[:space:]]*//' | sed 's#^"tx":\[##g' |
    sed 's#},{"time":#},\n{"time":#g' | grep '{"time"' >>
    $blockFile.aux

cat $blockFile.aux | while read line
do
$BASEDIR/script_save_transactions_to_file.sh "$line"
done

rm $blockFile.aux
```

Listing A.7: Script for taking transactions out of a block

```
#!/bin/bash

yesterday=`date +%Y/%m/%d --date=yesterday`
BASEDIR=$(dirname $0)

tx=$1

txHash=`echo $tx | sed 's#,#,\n#g' | grep "hash" | cut -d'"' -f4 |
    grep -v "^$" | head -n1`
echo $txHash
echo $tx | sed 's#,#,\n#g' > /tmp/aux_transaction_$txHash.txt

#Timestamp of the transaction
txTime=`cat /tmp/aux_transaction_$txHash.txt | grep "time" | cut -
    d":" -f2 | cut -d"," -f1`

#IP
txIP=`cat /tmp/aux_transaction_$txHash.txt | grep "relayed_by" |
    cut -d'"' -f4`

#Number of inputs
txVIN=`cat /tmp/aux_transaction_$txHash.txt | grep "vin_sz" | cut
    -d":" -f2 | cut -d"," -f1`

#Inputs
#HACER ALGO ESPECIAL
txInputs=`echo $tx | tr ']' '[' | cut -d"[" -f2`
#txInputs=`cat /tmp/aux_transaction_$txHash.txt | grep "inputs" |
    cut -d"[" -f2 | cut -d"]" -f1`

#Number of outputs
txVOUT=`cat /tmp/aux_transaction_$txHash.txt | grep "vout_sz" |
    cut -d":" -f2 | cut -d"," -f1`

#Outputs
#HACER ALGO ESPECIAL
txOut=`echo $tx | tr ']' '[' | cut -d"[" -f4`
#txOut=`cat /tmp/aux_transaction_$txHash.txt | grep "out" | cut -d
    "[" -f2 | cut -d"]" -f1`

#Version
txVersion=`cat /tmp/aux_transaction_$txHash.txt | grep "ver" | cut
     -d":" -f2 | cut -d"," -f1`

#Index
txIndex=`echo $tx | tr ']' '[' | cut -d"[" -f1,3,5 | sed 's#,#,\n#
    g' | grep "tx_index" | cut -d":" -f2 | cut -d"," -f1`

#Size
txSize=`cat /tmp/aux_transaction_$txHash.txt | grep "size" | cut -
    d":" -f2 | cut -d"," -f1`

#BlockHeight
txBlockHeight=`cat /tmp/aux_transaction_$txHash.txt | grep "
    block_height" | cut -d":" -f2 | cut -d"," -f1`

#Formato simple
#time;hash;IP;vIN;inputs;vout;out;version
echo $txTime";"$txHash";"$txIP";"$txVIN";"$txInputs";"$txVOUT";"
    $txOut";"$txVersion >> $BASEDIR/cuaderno/$yesterday/
    transacciones/simple.txt

#Formato completo
#time;hash;IP;vIN;inputs;vout;out;version;index;size;blockheight
echo $txTime";"$txHash";"$txIP";"$txVIN";"$txInputs";"$txVOUT";"
    $txOut";"$txVersion";"$txIndex";"$txSize";"$txtxBlockHeight >>
    $BASEDIR/cuaderno/$yesterday/transacciones/completa.txt

rm /tmp/aux_transaction_$txHash.txt
```

Listing A.8: Script for storing a transaction

A.5 Temporal correlations

For the analysis of temporal correlations two scripts have been developed, one calculates all the transactions which belong to each group and other one calculates the data needed for a chart representation.

In the first script only two parameters are needed, the user and password of the database.

```
#!/bin/bash

user=$1
pass=$2
BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_transaction_evolution
mkdir $BASEDIR/results_transaction_evolution_several_inputs
mkdir $BASEDIR/results_transaction_evolution_two_outputs
rm $BASEDIR/results_transaction_evolution/transactions.*
rm $BASEDIR/results_transaction_evolution_several_inputs/
    transactions_several_inputs.*
rm $BASEDIR/results_transaction_evolution_two_outputs/
    transactions_two_outputs.*
resultFile=`echo $BASEDIR"/results_transaction_evolution/
    transactions"`
resultFile2=`echo $BASEDIR"/
    results_transaction_evolution_several_inputs/
    transactions_several_inputs"`
resultFile3=`echo $BASEDIR"/
    results_transaction_evolution_two_outputs/
    transactions_two_outputs"`
logFile=`echo $BASEDIR"/results_transaction_evolution/logfile.log"
    `
errorFile=`echo $BASEDIR"/results_transaction_evolution/errorfile.
    log"`

echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile


initData=`date +%s --date="2009/01/01 00:00:00"`
echo $initData > /tmp/auxiliarEndData.txt

echo "2009
2010
2011
2012
2013" | while read year
do
    initData=`cat /tmp/auxiliarEndData.txt`
  echo "01
  02
  03
  04
  05
  06
  07
  08
  09
  10
  11
  12" | while read month
  do
    endData=`date +%s --date="$year/$month/01 00:00:00"`

    echo $initData $endData

    bloc=`echo $year"."$month`
    echo $bloc >> $errorFile
     if [ "$bloc" != "2009.01" ] && [ "$bloc" != "2013.05" ] && [ "
    $bloc" != "2013.06" ] && [ "$bloc" != "2013.07" ] && [ "$bloc"
    != "2013.08" ] && [ "$bloc" != "2013.09" ] && [ "$bloc" != "
    2013.10" ] && [ "$bloc" != "2013.11" ] && [ "$bloc" != "2013.12
    " ]
    then
      auxBlock1=`mysql -u $user --password="$pass" -e "select
    block_id from block where block_nTime>='$initData' and
    block_nTime<'$endData'" btg | grep -v "block_id" | sort | uniq`
      auxBlock2=`echo "$auxBlock1" | tr '\n' ','`
    auxsqlBlock=`echo "("${auxBlock2:0: -1}")"`

    auxTrans1=`mysql -u $user --password="$pass" -e "select tx_id from block_tx where block_id in $auxsqlBlock" btg | grep -v "tx_id" | sort | uniq`

    echo "$auxTrans1" > /tmp/evolution_transactions_file_aux.txt
    num=`wc -l /tmp/evolution_transactions_file_aux.txt | cut -d " " -f1`
    i=0
    while [ $num -gt 0 ]
    do
        wc -l /tmp/evolution_transactions_file_aux.txt
        auxTrans2=`head -n10000 /tmp/evolution_transactions_file_aux.txt | tr '\n' ','`
        auxsqlTrans=`echo "("${auxTrans2:0: -1}")"`

        mysql -u $user --password="$pass" -e "select tx_id, count(*) from txin where tx_id in $auxsqlTrans group by tx_id order by count(*) desc" btg | tr '\t' ';' > $resultFile.$year.$month.$i.txt
        mysql -u $user --password="$pass" -e "select tx_id, count(*) from txout where tx_id in $auxsqlTrans group by tx_id having count(*)=2" btg | tr '\t' ';' > $resultFile3.$year.$month.$i.txt
        sed '1,10000d' /tmp/evolution_transactions_file_aux.txt > /tmp/evolution_transactions_file_aux2.txt
        mv /tmp/evolution_transactions_file_aux2.txt /tmp/evolution_transactions_file_aux.txt
        num=`wc -l /tmp/evolution_transactions_file_aux.txt | cut -d" " -f1`
        let i=$i+1
    done
    cat $resultFile.$year.$month.*.txt | awk -F';' '{print $2";"$1}' | grep -v "count" | sort -nr > $resultFile.$year.$month.txt
    cat $resultFile3.$year.$month.*.txt | awk -F';' '{print $2";"$1}' | grep -v "count" | sort -nr > $resultFile3.$year.$month.txt
    cat $resultFile.$year.$month.*.txt | awk -F';' '{print $2";"$1}' | grep -v "count" | grep -v "^1;" | sort -nr > $resultFile2.$year.$month.txt
    fi

    echo $endData > /tmp/auxiliarEndData.txt
    initData=`cat /tmp/auxiliarEndData.txt`
  done
done >> $logFile 2>> $errorFile

echo "DATE END: `date`" >> $logFile
```

Listing A.9: Script for temporal correlations - Calculate transactions

```
#!/bin/bash

BASEDIR=$(dirname $0)
mkdir $BASEDIR/results_transaction_evolution_statistics
resultFile=`echo $BASEDIR"/results_transaction_evolution/
    transactions"`
resultFile2=`echo $BASEDIR"/
    results_transaction_evolution_several_inputs/
    transactions_several_inputs"`
resultFile3=`echo $BASEDIR"/
    results_transaction_evolution_two_outputs/
    transactions_two_outputs"`
fileAll=`echo $BASEDIR"/results_transaction_evolution_statistics/
    transactions_all_statistics.txt"`
fileInput=`echo $BASEDIR"/results_transaction_evolution_statistics
    /transactions_input_statistics.txt"`
fileOutput=`echo $BASEDIR"/
    results_transaction_evolution_statistics/
    transactions_output_statistics.txt"`
logFile=`echo $BASEDIR"/results_transaction_evolution_statistics/
    logfile.log"`
errorFile=`echo $BASEDIR"/results_transaction_evolution_statistics
    /errorfile.log"`

rm $fileAll
rm $fileInput
rm $fileOutput
touch $fileAll
touch $fileInput
touch $fileOutput

echo "DATE INIT: `date`" > $logFile
echo "DATE INIT: `date`" > $errorFile


initData=`date +%s --date="2009/01/01 00:00:00"`
echo $initData > /tmp/auxiliarEndData.txt

echo "2009
2010
2011
2012
2013" | while read year
do
  initData=`cat /tmp/auxiliarEndData.txt`
  echo "01
  02
  03
  04
  05
  06
  07
  08
  09
  10
  11
  12" | while read month
  do
    endData=`date +%s --date="$year/$month/01 00:00:00"`

    echo $initData $endData

    bloc=`echo $year"."$month`
    echo $bloc >> $errorFile
    if [ "$bloc" != "2009.01" ] && [ "$bloc" != "2013.05" ] && [ "$bloc" != "2013.06" ] && [ "$bloc" != "2013.07" ] && [ "$bloc" != "2013.08" ] && [ "$bloc" != "2013.09" ] && [ "$bloc" != "2013.10" ] && [ "$bloc" != "2013.11" ] && [ "$bloc" != "2013.12" ]
    then
      allTrans=`wc -l $resultFile.$year.$month.txt | cut -d" " -f1`
      inputTrans=`wc -l $resultFile2.$year.$month.txt | cut -d" " -f1`
```

```
        outputTrans=`wc -l $resultFile3.$year.$month.txt | cut -d" "
      -f1`
        echo $year"-"$month";"$allTrans >> $fileAll
        echo $year"-"$month";"$inputTrans >> $fileInput
        echo $year"-"$month";"$outputTrans >> $fileOutput
      fi

      echo $endData > /tmp/auxiliarEndData.txt
      initData=`cat /tmp/auxiliarEndData.txt`
    done
done >> $logFile 2>> $errorFile

echo "DATE END: `date`" >> $logFile
```

Listing A.10: Script for temporal correlations - Present data

A.6 Anonymity of users

Several scripts have been developed for searching in Bitcointalk, pastebin and bitbin interesting information related with Bitcoin or to detect Bitcoin addresses. They have to be executed every 2 minutes in order to constantly download the latests information, for doing this the best option is to use crontab.

```
#!/bin/bash

today=`date +%Y/%m/%d`
rand=`echo $RANDOM`
BASEDIR=$(dirname $0)
mkdir -p $BASEDIR/cuaderno/$today/scrapers

tail -n100 $BASEDIR/listado_urls_bitcointalk.txt > $BASEDIR/
    listado_urls_bitcointalk.aux
mv $BASEDIR/listado_urls_bitcointalk.aux $BASEDIR/
    listado_urls_bitcointalk.txt

wget -q -O /tmp/bitcointalk_auxfile1.txt.$rand https://bitcointalk
    .org/index.php?action=recent
```

```
egrep -o "<a href=\"https://bitcointalk.org/index.php\?topic=[^\"]*\"" /tmp/bitcointalk_auxfile1.txt.$rand | egrep -o "https://bitcointalk.org/index.php\?topic=[^\"]*" | while read url
do
  post=`echo $url | egrep -o "https://bitcointalk.org/index.php\?topic=[0-9]*"`
  getUrl=`grep "$url" $BASEDIR/listado_urls_bitcointalk.txt`
  findUrl=`echo $?`
  if [[ $findUrl -ne 0 ]]
  then
    echo $url >> $BASEDIR/listado_urls_bitcointalk.txt
    allURL=`echo $url".0;all"`
    wget -q -O /tmp/bitcointalk_auxfile2.txt.$rand $allURL

    egrep -n -o "[^0-9A-Za-z][13][1-9A-HJ-NP-Za-km-z]{26,33}[^0-9A-Za-z]" /tmp/bitcointalk_auxfile2.txt.$rand | while read address
    do
      line=`echo $address | cut -d":" -f1`
      bitcoinAddress=`echo $address | egrep -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}"`
      user=`tac /tmp/bitcointalk_auxfile2.txt.$rand | tail -n$line | grep "profile of" | head -n1 | sed 's/.*View the profile of [^>]*//g' | cut -d">" -f2 | cut -d"<" -f1`
      if [ -n "$user" ]
      then
        echo $bitcoinAddress";"$user";"$post
      fi
    done | sort | uniq > /tmp/bitcointalk_auxfile3.txt.$rand

    auxAddress=`cat /tmp/bitcointalk_auxfile3.txt.$rand | cut -d";" -f1,2 | sort | uniq | cut -d";" -f1 | sort | uniq -c | sort -nr | grep "^ *1 "`
    egrep "^$auxAddress$" /tmp/bitcointalk_auxfile3.txt.$rand | tee -a $BASEDIR/cuaderno/$today/scrapers/bitcointalk_BitcoinAddress_limit.txt
    cat /tmp/bitcointalk_auxfile3.txt.$rand | cut -d";" -f1,2 | sort | uniq | cut -d";" -f1 | sort | uniq -c | sort -nr | grep -v "^ *1 " | awk '{print $2}' | while read line
    do
      grep "^$line;" /tmp/bitcointalk_auxfile3.txt.$rand | head -n1
```

```
 done | tee -a $BASEDIR/cuaderno/$today/scrapers/bitcointalk_BitcoinAddress_limit.txt

 egrep -n -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}" /tmp/bitcointalk_auxfile2.txt.$rand | while read address
 do
   line=`echo $address | cut -d":" -f1`
   bitcoinAddress=`echo $address | egrep -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}"`
   user=`tac /tmp/bitcointalk_auxfile2.txt.$rand | tail -n$line | grep "profile of" | head -n1 | sed 's/.*View the profile of [^>]*//g' | cut -d">" -f2 | cut -d"<" -f1`
   if [ -n "$user" ]
   then
     echo $bitcoinAddress";"$user";"$post
   fi

   echo "#####" >> $BASEDIR/bitcointalk.log
   echo $bitcoinAddress";"$user";"$post";"$url >> $BASEDIR/bitcointalk.log
   egrep -n -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}" /tmp/bitcointalk_auxfile2.txt.$rand >> $BASEDIR/bitcointalk.log
   echo "#####" >> $BASEDIR/bitcointalk.log
 done | sort | uniq > /tmp/bitcointalk_auxfile3.txt.$rand

 auxAddress=`cat /tmp/bitcointalk_auxfile3.txt.$rand | cut -d";" -f1,2 | sort | uniq | cut -d";" -f1 | sort | uniq -c | sort -nr | grep "^ *1 "`
 egrep "^$auxAddress$" /tmp/bitcointalk_auxfile3.txt.$rand | tee -a $BASEDIR/cuaderno/$today/scrapers/bitcointalk_BitcoinAddress_all.txt
 cat /tmp/bitcointalk_auxfile3.txt.$rand | cut -d";" -f1,2 | sort | uniq | cut -d";" -f1 | sort | uniq -c | sort -nr | grep -v "^ *1 " | awk '{print $2}' | while read line
 do
   grep "^$line;" /tmp/bitcointalk_auxfile3.txt.$rand | head -n1
 done >> $BASEDIR/cuaderno/$today/scrapers/bitcointalk_BitcoinAddress_all.txt

 rm /tmp/bitcointalk_auxfile2.txt.$rand
 rm /tmp/bitcointalk_auxfile3.txt.$rand
```

```
  fi
done | cut -d";" -f1-2 | sort | uniq >>  $BASEDIR/cuaderno/$today/
    scrapers/bitcointalk_BitcoinAddress_users_limit.txt

cat $BASEDIR/cuaderno/$today/scrapers/
    bitcointalk_BitcoinAddress_users_limit.txt | sort | uniq >
    $BASEDIR/cuaderno/$today/scrapers/
    bitcointalk_BitcoinAddress_users_limit.aux

mv $BASEDIR/cuaderno/$today/scrapers/
    bitcointalk_BitcoinAddress_users_limit.aux $BASEDIR/cuaderno/
    $today/scrapers/bitcointalk_BitcoinAddress_users_limit.txt

rm /tmp/bitcointalk_auxfile1.txt.$rand
```

Listing A.11: Script for downloading the addresses found in the lasts posts of Bitcointalk

```
#!/bin/bash

today=`date +%Y/%m/%d`
rand=`echo $RANDOM`
BASEDIR=$(dirname $0)
mkdir -p $BASEDIR/cuaderno/$today/scrapers

tail -n1000 $BASEDIR/listado_urls_bitbin.txt > $BASEDIR/
    listado_urls_bitbin.aux
mv $BASEDIR/listado_urls_bitbin.aux $BASEDIR/listado_urls_bitbin.
    txt

wget -q -O /tmp/bitbin_auxfile1.txt.$rand  http://bitbin.it/
    latest_pastes.php

egrep -o "http://bitbin.it/[a-zA-Z0-9]{8}" /tmp/bitbin_auxfile1.
    txt.$rand | while read file
do
  getUrl=`grep "$file" $BASEDIR/listado_urls_bitbin.txt`
  findUrl=`echo $?`
  if [[ $findUrl -ne 0 ]]
  then
    echo $file >> $BASEDIR/listado_urls_bitbin.txt
    wget -q -O /tmp/bitbin_auxfile2.txt.$rand $file
```

```
found=`cat /tmp/bitbin_auxfile2.txt.$rand | egrep -o "[^0-9A-Za-z][13][1-9A-HJ-NP-Za-km-z]{26,33}[^0-9A-Za-z]" | grep -v "193MHato6vuRXFBm45FnSnFTXeyxpByaSC" | egrep -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}"`
result=`echo $?`
auxiliar=`echo "$found" | sort | uniq`
if [[ $result -eq 0 ]]
then
  echo "$auxiliar" | while read line
  do
    echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/bitbin_BitcoinAddress_limit.txt
  done
fi

found=`cat /tmp/bitbin_auxfile2.txt.$rand | egrep -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}" | grep -v "193MHato6vuRXFBm45FnSnFTXeyxpByaSC" | egrep -o "[13][1-9A-HJ-NP-Za-km-z]{26,33}"`
result=`echo $?`
auxiliar=`echo "$found" | sort | uniq`
if [[ $result -eq 0 ]]
then
  echo "$auxiliar" | while read line
  do
    echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/bitbin_BitcoinAddress_all.txt
  done
fi

            found=`cat /tmp/bitbin_auxfile2.txt.$rand | grep -v "<meta name=\"keywords\"" | egrep -i "bitcoin"`
            result=`echo $?`
            auxiliar=`echo "$found" | sort | uniq`
            if [[ $result -eq 0 ]]
            then
                    echo "$auxiliar" | while read line
                    do
                            echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/bitbin_general.txt
                    done
            fi
    rm /tmp/bitbin_auxfile2.txt.$rand
        fi
done

rm /tmp/bitbin_auxfile1.txt.$rand
```

Listing A.12: Script for downloading the addresses or information related to Bitcoin in Bitbin

```
#!/bin/bash

today=`date +%Y/%m/%d`
rand=`echo $RANDOM`
BASEDIR=$(dirname $0)
mkdir -p $BASEDIR/cuaderno/$today/scrapers

tail -n1000 $BASEDIR/listado_urls_pastebin.txt > $BASEDIR/
    listado_urls_pastebin.aux
mv $BASEDIR/listado_urls_pastebin.aux $BASEDIR/
    listado_urls_pastebin.txt

numProxies=`wc -l $BASEDIR/proxy_list_2.txt | cut -d" " -f1`
lineProxies=`echo $(( $RANDOM%$numProxies ))`
proxy=`awk "NR==$lineProxies" $BASEDIR/proxy_list_2.txt`
export http_proxy="http://$proxy"
wget -q --tries=2 --referer="http://www.google.com" --user-agent="
    Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.6)
    Gecko/20070725 Firefox/2.0.0.6" --header="Accept: text/xml,
    application/xml,application/xhtml+xml,text/html;q=0.9,text/
    plain;q=0.8,image/png,*/*;q=0.5" --header="Accept-Language: en-
    us,en;q=0.5" --header="Accept-Encoding: gzip,deflate" --header=
    "Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7" --header="Keep
    -Alive: 300" -dnv -O /tmp/pastebin_auxfile1.txt.$rand http://
    pastebin.com/archive

wc -l /tmp/pastebin_auxfile1.txt.$rand

egrep -o "a href=\"/[a-zA-Z0-9]{8}\"" /tmp/pastebin_auxfile1.txt.
    $rand | cut -d'"' -f2 | while read auxfile
do
  file=`echo "http://pastebin.com"$auxfile`
  getUrl=`grep "$file" $BASEDIR/listado_urls_pastebin.txt`
```

```
    findUrl=`echo $?`
    if [[ $findUrl -ne 0 ]]
    then
      echo $file >> $BASEDIR/listado_urls_pastebin.txt
      wget -q --tries=2 --referer="http://www.google.com" --user-
     agent="Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv
     :1.8.1.6) Gecko/20070725 Firefox/2.0.0.6" --header="Accept:
     text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,
     text/plain;q=0.8,image/png,*/*;q=0.5" --header="Accept-Language
     : en-us,en;q=0.5" --header="Accept-Encoding: gzip,deflate" --
     header="Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7" --
     header="Keep-Alive: 300" -dnv -O /tmp/pastebin_auxfile2.txt.
     $rand $file

wc -l /tmp/pastebin_auxfile2.txt.$rand

      found=`cat /tmp/pastebin_auxfile2.txt.$rand | egrep -o "[^0-9A
     -Za-z][13][1-9A-HJ-NP-Za-km-z]{26,33}[^0-9A-Za-z]"`
      result=`echo $?`
      auxiliar=`echo "$found" | egrep -o "[13][1-9A-HJ-NP-Za-km-z
     ]{26,33}" | sort | uniq`
      if [[ $result -eq 0 ]]
      then
        echo "$auxiliar" | while read line
        do
          echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/
     pastebin_BitcoinAddress_limit.txt
        done
      fi

      found=`cat /tmp/pastebin_auxfile2.txt.$rand | egrep -o "
     [13][1-9A-HJ-NP-Za-km-z]{26,33}"`
      result=`echo $?`
      auxiliar=`echo "$found" | sort | uniq`
      if [[ $result -eq 0 ]]
      then
        echo "$auxiliar" | while read line
        do
          echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/
     pastebin_BitcoinAddress_all.txt
        done
      fi
```

```

            found=`cat /tmp/pastebin_auxfile2.txt.$rand | egrep -i "bitcoin"`
            result=`echo $?`
            auxiliar=`echo "$found" | sort | uniq`
            if [[ $result -eq 0 ]]
            then
                  echo "$auxiliar" | while read line
                  do
                        echo $line";"$file >> $BASEDIR/cuaderno/$today/scrapers/pastebin_general.txt
                  done
            fi

   rm /tmp/pastebin_auxfile2.txt.$rand
  fi
done

unset http_proxy

rm /tmp/pastebin_auxfile1.txt.$rand
```

Listing A.13: Script for downloading the addresses or information related to Bitcoin in Pastebin

Appendix B

Bitcoin Visualizer

At the start of the project the idea was to use the Bitcoin Visualizer [17] application to study the aggregation methods that it uses and improve them, due to the problems encountered it was later decided not to use it, and implement in another way the aggregations studied.

From the beginning there were problems compiling the jar with the files updated so I contacted the author to be able to resolve this problems and work with the application.

After a lot of help from him we were able to init the application and begin downloading the block chain. The changes done were:

- Get the jar file directly from the author, as well as a folder with libraries and a certificate.
- Change the neo4j.properties file by the one obtained from the author.
- When running the application, execute first the Neo4jCoordinator application.
- Adding to the installation folder a file named 1.json that represents the first block to be able to begin the download of the rests of the blocks.
- Change the password of root in the mysql installation.

- Use a different command when running the application the first time: java -jar -Xmx2560m D:\ProgramData\BlockViewer\BlockViewer.jar -dbPath C:\Bitcoin\neo4j-enterprise-1.8.2\data\graph.db -configPath C:\Bitcoin\neo4j-enterprise-1.8.2\conf\neo4j.properties -validate true -low -high -scraper

After all this changes and modifications I was able of downloading the block chain in json files, which had at the moment a size of around 67GB. But when trying to run the aggregations from the application the process continued failing with NullPointerException errors.

Seeing that the time was quickly passing and I was not able to advance it was decided to abandon it and continue with my own programs.