University of Pisa
Department of Computer Science
PhD in Computer Science - XXX cycle

Ph.D. Thesis

Blockchain Applications: Bitcoin and Beyond

Damiano Di Francesco Maesa
email: d.difrancesco@for.unipi.it

Supervisor:
Prof. Laura Ricci
University of Pisa

Supervisor:
Dr. Andrea Marino
University of Pisa

October - 2018

Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Blockchain Applications: Bitcoin and Beyond

Damiano Di Francesco Maesa

Supervisor

Prof. Laura Ricci
Dott. Andrea Marino

October 2018

Author's e-mail: d.difrancesco@for.unipi.it

Author's address:

Dipartimento di Informatica
Università degli Studi di Pisa
Largo Bruno Pontecorvo, 3
56127 Pisa
Italia

A tutte le Laure che hanno reso possibile
arrivare fin qui ed alla bella bimba che mi
ha accompagnato al traguardo.

Abstract

During the last years a novel technology, called blockchain, has gained more and more momentum, attracting the interest of big corporations and governments alike. Despite still being in its infancy, this technology has already provided a long lived usage example through the Bitcoin cryptocurrency and more novel application proposals keep getting presented. As the success of Bitcoin proved the feasibility of cryptocurrencies to provide a new trustless value exchange paradigm, the application of blockchain technology to other fields promises to revolutionize how we manage information in general, removing the need for trusted central authorities and intermediaries by giving control back to the users.

In this dissertation we present our contribution to the field. Such contribution is twofold and can be separated in two conceptually distinct lines of work. First, we focused our attention on the Bitcoin cryptocurrency, and in particular on its *users graph*, where nodes represent users taking part in the system and edges correspond to transactions between them. We conducted a vast set of analysis on the data contained in the Bitcoin blockchain finding peculiar behaviours in the corresponding network and observing special sets of transactions. We gained useful insights on both the macroscopic state of the system and particular users use cases. Not only the gathered information is interesting by itself, but it is also useful in assessing the real users privacy provided by the Bitcoin protocol.

The second main line of work presented in this dissertation concerns the novel applications of blockchain technology in fields other than cryptocurrencies. We present our own novel contribution of integrating blockchain with Access Control systems. We present different levels of possible integration, showing the gained advantages and drawbacks for each one, always relying on real proof of concept implementations of our proposals. This allows us to evaluate what true benefits the properties provided by blockchain technology can bring to traditionally centralised systems.

Acknowledgments

The author would like to thank Dr. Paolo Mori of the Institute of Informatics and Telematics (IIT) at the Italian National Research Centre (CNR) in Pisa for his help and collaboration during the author's PhD years.

I would also like to thank Dr. Christian Decker and Prof. Roger Wattenhofer of the Distributed Computing Group of the ETH Zürich for hosting me at their lab and helping me with my research during my period as visiting academic guest at the ETH Zürich.

Finally, I would like to express my gratitude to everyone that helped me during the writing of this thesis and the four years of my PhD. In particular, I thank Prof. Alexey Vinel of the School of Information Technology at Halmstad University (Sweden) and Prof. Jose Antonio Fernandes de Macedo of the Computer Science Department of the Federal University of Cear (Brazil) for their helpful reviews of this thesis.

Contents

List of Figures

List of Tables

List of Publications

In International Journals

[82] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "Data-driven analysis of Bitcoin properties: exploiting the users graph". *International Journal of Data Science and Analytics (2017)*: 1-18.

[83] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "Detecting Artificial Behaviours in the Bitcoin Users Graph". *Online Social Networks and Media 3 (2017)*: 63-74.

In International Conferences

[81] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "Uncovering the bitcoin blockchain: an analysis of the full users graph." In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pp. 537-546. IEEE, 2016.

[80] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "An analysis of the Bitcoin users graph: inferring unusual behaviours." In *Complex Networks and their Applications*, pp. 749-760. Springer, Cham, 2016.

[84] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "Blockchain based access control." In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 206-220. Springer, Cham, 2017.

[79] Damiano Di Francesco Maesa, Matteo Franceschi, Barbara Guidi, and Laura Ricci. "BITKER: a P2P kernel client for Bitcoin". In *16th International Conference on High Performance Computing & Simulation (HPCS 2018)*.

[85] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "Blockchain Based Access Control Services." In *IEEE Symposium on Recent Advances on Blockchain and its Applications, 2018 IEEE International Conference on Blockchain*, 2018, to appear.

Technical Reports

[86] Damiano Di Francesco Maesa, Laura Ricci, and Paolo Mori. "Distributed Access Control Through Blockchain Technology." *Blockchain Engineering (2017)*:

31, Ercim News.

[78] Damiano Di Francesco Maesa. "Bitcoin Protocol Main Threats." Technical Report, Computer Science Department, TR . University of Pisa, Pisa, IT, 2017. http://eprints.adm.unipi.it/2371/1/TechRep.pdf

Submitted Papers

[-] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "A blockchain based approach for the definition of auditable access control systems". Journal, under review.

[-] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "The Graph Structure of Bitcoin". Conference, under review.

Introduction

Conceptually, a blockchain is a shared, immutable and tamper resistant collection of records, which is updated in a *peer to peer* decentralized way through a distributed consensus algorithm. Such collection is stored by each peer, that locally keeps its copy consistent with the agreed current version. New participants can freely join the system and start participating in the maintenance of the database through the consensus protocol. Since there is no central point of control, there is no central point of failure either, all users share the same privileges. This means that the resulting system is truly democratic and it can not be easily controlled, manipulated or shut down.

The main technological innovation introduced by blockchain technology is its ability to create a trusted environment for reciprocally untrusted entities to collaborate (to the management of a common information repository), without the need for a central authority or intermediaries. There is no need for trust between users, since trust is established by the technology itself. A blockchain gives entities the freedom to exchange data without the need for any kind of external supervision. This is technically achieved employing a distributed consensus algorithm, which informally moves the concept of trust from the individual level to the community level, i.e. it allows for a trusted community of untrusted individuals. The disruptive aspect of blockchain is exactly the property of creating trust at a technology level, this gained it the name of *trust machine* [258].

Such a technology can be used both by companies wishing to be more transparent and accountable, as well as the ones wishing to cut costly trusted third party intermediaries. In fact the traditional task of ensuring trust is taken over by the technological layer. Cutting the need for intermediaries results in less costs, transparency and greater efficiency. Furthermore, the integration of blockchain with cryptography allows for proof of ownership and authorship of information, while its nature of digital asset allows to embed executable code as data, allowing the usage of smart contracts. This allows for a wide range of applications well beyond the first use case of Bitcoin in particular and cryptocurrencies in general, e.g. electronic voting, identity management systems, health care records, distributed notaries, intellectual property protection supply chain management, as shown in Chapter 5.

Blockchain Spread. The interest around distributed ledgers in general, and blockchain technology in particular, has sprung during the last two years. Until 2014 it was extremely difficult to find a single academic paper about blockchain technology that was not either about Bitcoin or completely speculative. Before that year, the terms Bitcoin and Blockchain were intertwined and often used interchangeably among the broader public.

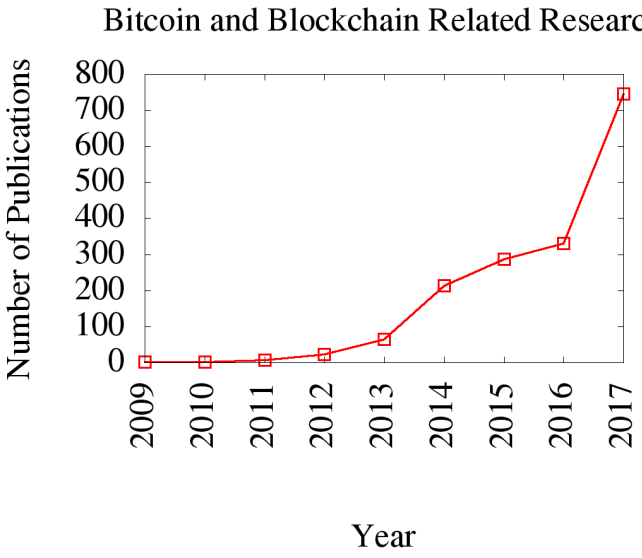


Figure 1: Bitcoin and Blockchain related published research by publication year. Data source [72].

The term *blockchain* itself was not born with Bitcoin, since there is no presence of it in the Bitcoin whitepaper [195]. Instead it is believed that the term, with its intended meaning, was first used during November of 2008 on the cryptography mailing list where Bitcoin early enthusiast used to discuss with Bitcoin creator Satoshi Nakamoto [225]. The concept of blockchain was nevertheless first introduced to manage the Bitcoin public transaction ledger and this is why it is only one concept of a greater discipline called *Distributed Ledger Technology*.

In [281] the authors analyse the academic publications about blockchain until 2015 (included), and found that about 80% of them were explicitly about Bitcoin. Bitcoin advocates were predicting that blockchain technology would have changed our society forever, but few ears were listening. But 2014 saw the birth of the Ethereum project, even if there was no public version of Ethereum until 2015, the unspoken possibilities introduced by real smart contracts showed how blockchain technology could reach far beyond cryptocurrency applications. The concept of Blockchain 3.0 was born. The great price rallies and the sudden appearance of Bitcoin millionaires brought the public attention to the matter, and soon the big companies followed suit, trying to take advantage of *the biggest revolution since the Internet* [277]. This was true not only about companies, but about the academia as well. [72] provides a database of Bitcoin and Blockchain related published research. Plotting the entries by publication year we obtain the graph in Figure 1. The trend is clear from the figure, even if still a lot of the considered publications concerned mostly Bitcoin. In [107] the authors analyse the GitHub projects about blockchain technology. As reported in Figure 2, the number of projects is clearly raising, in-

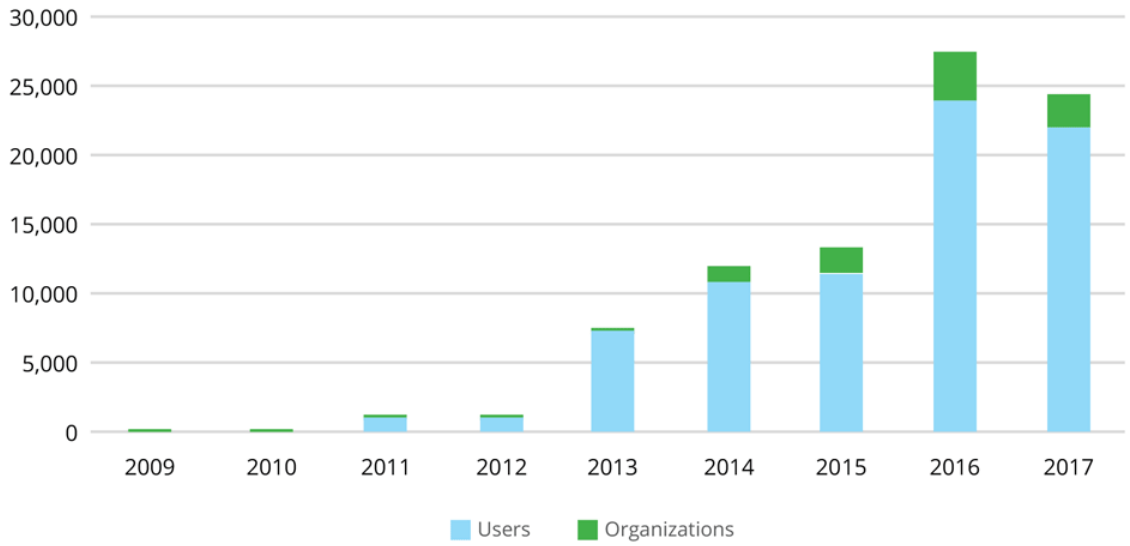


Figure 2: New GitHub projects concerning blockchain technology between 2009 and 2017 (first half only). Source [107].

cluding the contribution given by organizations. In the same study the authors also point out that the ecosystem is highly dynamic, with only 8% of projects actively maintained, and projects having in general a 1.22 years life span.

The spread of blockchain technology is also reflected by the fast growing investments made in new token launches, the so called *Initial Coin Offering*, or *ICO*. An ICO is a novel, cryptocurrencies based, kind of crowdfunding, used to finance the launch of a new blockchain project, rewarding financiers with tokens that will be valuable inside the project, if it succeeds. A study of the funds raised in the last years is presented in [117]. According to [62], ICOs have raised as much as $ 95 181 391 in 2016, $ 3 880 018 203 in 2017 and a staggering $ 6 175 256 332 during the first four months of 2018 alone. So, it is not surprising that blockchain technology made its way into the prestigious research and advisory firm Gartner, *Top Trends in the Gartner Hype Cycle for Emerging Technologies, 2017* [262].

Distributed Ledger Technology A *distributed ledger* [266] (often referred as *DLT*, from Distributed Ledger Technology) is a decentralised repository of data managed and maintained by many participants, without necessity of assuming trust between each other. In general, the participants have the same rights and control over the repository, and communicate directly between themselves in a P2P fashion to propose and notify updates of such repository. Often such updates satisfy an append only rule to guarantee the data immutability property. There is no need for intermediaries nor for a centralised controller, since the participants employ a distributed consensus algorithm to reach a decision on the updates to be made to the repository.

Even if it is called *distributed* ledger it would be technically more precise to call it *decentralised* ledger. In fact, in computer science the term *distributed* is mainly used to indicate a network of autonomous entities communicating between themselves to reach a common goal, like a distributed ledger, but, in general, no coherence is assumed between the participants. Each node could be executing a completely different task while communicating its result as data for the other nodes. For example, a distributed system can be employed to coordinate a set of threads computing different algorithms. In a distributed ledger, instead, each (compliant) node is expected to follow the same protocol to reach the same result (each time a consensus is reached), i.e. each honest participant should end up with the same copy of the repository. In fact there is no unique agreed ledger, instead a copy of the same ledger is stored and maintained by each node, and the ledger held by the majority of the network (non necessarily a numerical majority) is considered as the correct one. Formally, it is more correct to say that the ledger is *replicated*, since a copy of it is stored by each participant and the same management operations are repeated by all of them locally.

Blockchain technology is just one possible technology to implement a distributed ledger. A blockchain implements a distributed ledger by grouping records (i.e. ledger state updates) into blocks that are made tamper resistant by adding a cryptographic signature of the block data. Usually this is achieved by adding a cryptographic hash of the entire block content in the block header. The blocks are than chained together by back-linking each block to the predecessor in a tamper resistant way. Again, the most common way of achieving this is through cryptographic hash functions, by adding the hash of the previous block in the following block header. By making each block recursively dependent on both its content and the previous block in the chain, such block becomes dependent on the entire content of all the blocks before it, all the way to first block created (often called the *genesis* block). This way it is not possible to modify any data inside a block without invalidating all the subsequent blocks.

Implementing a distributed ledger with a blockchain allows to build an immutable, distributed, always available, secure and publicly accessible repository of data. Often, the records stored in each block are called *transactions* due to historical reasons. In fact, blockchain technology was first introduced to support cryptocurrencies (by the Bitcoin cryptocurrency protocol [195]) and in such a scenario the blockchain is used as a public ledger to store transfers of value between entities, called transactions.

The main issues with blockchain implementation of distributed ledgers are scalability and efficiency. A pure blockchain often uses expensive distributed consensus algorithms to guarantee an eventual consensus on the repository consistency in a trustless environment. But more efficient and simpler consensus algorithms are possible if we relax the trust assumptions in the system. In general, the more trust we place on entities and the more efficient the system gets, but often also more centralised. The different types of blockchains basically differ for the trust level

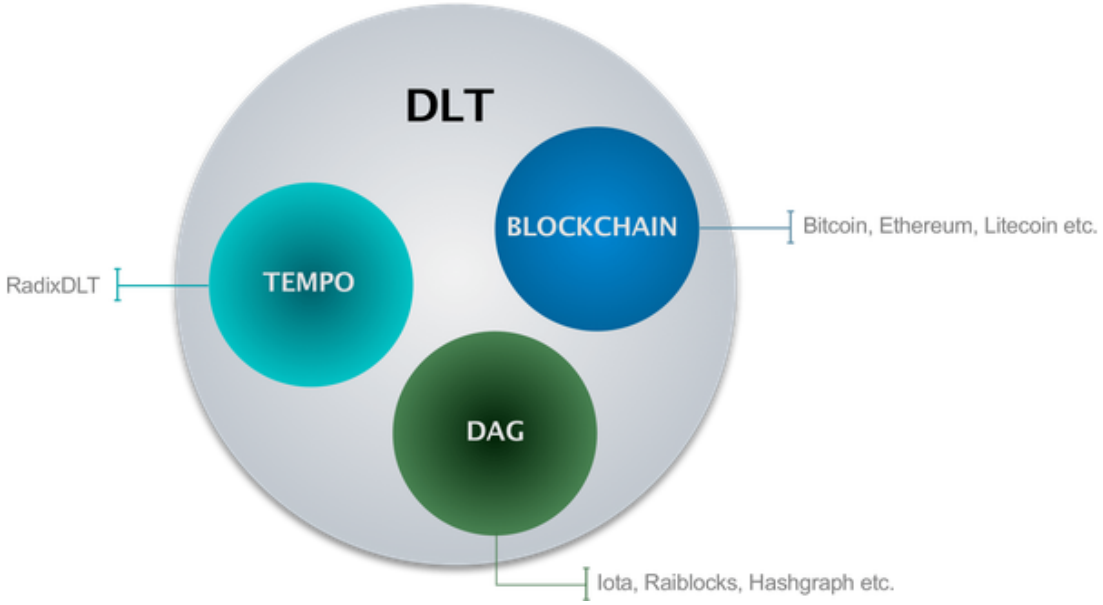


Figure 3: Distributed ledgers implementation proposals topology.

associated with *write* and *read* operations. By write operation we mean the ability to update the ledger, i.e. write content on it, while by read operation we mean the ability to read the blockchain content. Blockchains are called public (resp. private) whether any trustless (resp. only trusted) entities can read. They are called permissionless (resp. permissioned) whether any trustless (resp. only trusted) entities can write.

A blockchain can contain any type of data in its records, also code. But storing and executing is not the same thing. Coupling executable code with blockchain technology allows for the so called *smart contracts*. The term *contract* in the name can be misinterpreted, a smart contract is simply code containing arbitrary programming logic, not necessary a contract between entities. A smart contract supporting blockchain is a blockchain where the distributed consensus validates also the execution of the code contained in each block. Basically, each function call to the code repository stored in the blockchain is executed sequentially in the current block state, and the final state is updated accordingly. Given a block, each participant can re-execute the function calls it contains and check if the results are correct, i.e. the same they obtained. Executing a smart contract in the blockchain guarantees it a set of new properties, like:

- *atomicity*, an operation runs entirely or fails without affecting the state;
- *synchronicity*, the code is executed in a synchronous way;
- *provenance*, the code can only be executed by traceable external calls;

- *availability*, the code and associated data is always available;
- *immutability*, the code can not be changed or tampered with after deployment;
- *immortality*, the code and data can only be removed if it commits a self destruct operation.

It is said that the code execution is decentralized, or *performed by the blockchain*, but in practice it is replicated by all the participants. We will present a practical example of blockchain protocol supporting smart contracts in Section 1.6.

We do remark that blockchain is just a possible implementation of a distributed ledger, not the only one. For example, distributed ledgers not implemented with blockchains are Radix [224], IOTA [145], Hedera Hasgraph [132] and R3 Corda [223], see Figure 3.

I.1 Thesis Contribution

This thesis is focused on the study of two applications of Blockchain technology: Bitcoin and Access Control systems. In such a context our contribution is twofold.

The first contribution is the analysis of the *users graph* associated to the Bitcoin blockchain, as this cryptocurrency has been the first application of blockchain technology. The work done to perform this analysis allowed us to gain an useful insight about the underlying protocol and how it was exploited by users. The information gained can be especially useful to study the level of privacy provided by the Bitcoin protocol, and other pseudonymity based cryptocurrencies in general.

Taking advantage of this knowledge, the second contribution goes beyond Bitcoin, and cryptocurrencies in general, particular application, studying whether an application field, such as Access Control systems, can benefit from the integration of blockchain technology. This application is novel and our results are promising since they answer positively to this question.

To reflect these two different orthogonal contributions, the thesis is consequently divided in two parts.

Bitcoin Users Graph Analysis.

The automatic analysis and classification of interesting transaction patterns on a blockchain is an useful research topic that can lead to great understanding on how users interact within a cryptocurrency such as Bitcoin.

The main goal of the first part of this thesis is to study the Bitcoin users graph. The analysis performed gave us a good insight on the users behaviours behind the Bitcoin cryptocurrency, information that can be used to help asses the provided users privacy. Conceptually, in this graph there is one node for each Bitcoin user, and there is one edge from a node a to a node b whether there has been a transaction

from user a to user b. An user can own several addresses but this information is not publicly accessible.

The proposed graph perspective is one of the possible ways of modeling the hypergraph structure of Bitcoin. Indeed, each transaction in Bitcoin can be seen as a weighted hyperedge from a set of addresses directed to another set of addresses. The resulting weighted directed hypergraph is called the *transactions graph* (despite actually being an hypergraph) [82]. The users graph model we adopted to describe users activity allows us to borrow algorithms and tools from a vast literature provided by graph theory and network analysis.

In the Bitcoin protocol addresses corresponds to random strings, as pseudonyms, and all addresses activities are stored in the blockchain, which is a publicly readable ledger. In this sense, Bitcoin does not offer strong users anonymity, as it relies instead on the use of pseudonyms (see Section 1.2.1).

Collapsing together all the addresses belonging to the same user, into a single super node (named *cluster*) would give us the so called *users graph* from the transactions hypergraph. Being able to build the users graph effectively means breaking address pseudonymity. For this task, a set of deanonymization attacks trying to break addresses pseudonymity have been presented in the literature starting from the data contained inside the blockchain (see Section 1.5). All in all, there is no known exact method to achieve this but several approximated methods based on heuristic rules have been proposed in the literature (see Section 1.5.2). The heuristic we adopted is well-known [113] and it is based on the recursive application of a common sense rule: all the addresses belonging to a tail of an hyperedge belong to the same user, as, otherwise, different users would have to jointly sign the transaction due to the protocol rules. This heuristic tends to be conservative in the sense that it tries to minimize the number of false positive at the price of a less clustered resulting users graph. We built our Bitcoin users graph from the raw data contained in the Bitcoin blockchain using this rule and proposing a clustering algorithm linear in the sum of the number of addresses and transactions in the blockchain. This efficiency is necessary, since the Bitcoin blockchain has reached a size in the order of hundreds of Gigabytes.

Using this data, we evaluated the distribution of clusters sizes and isolated the biggest clusters, that unsurprisingly were identifiable as famous services in the Bitcoin economy.

We performed a wide set of classical graph analysis on our users graph. We first studied the evolution of the graph measurements over time. To this aim we studied the connectivity, densification, degree distributions and clustering coefficient of the graph over time. Through those analysis we were able to find a novel result. In fact, the relatively high clustering coefficient together with the low average distance suggested a small world phenomenon, but, surprisingly, the diameter was found to be high and constant. We also noticed evident outliers in both indegree and outdegree distributions of the complete graph.

We derived the list of central nodes using several different centrality definition.

Most measures returned, as most central, clusters linked to famous services in the Bitcoin economy. The ability to identify hubs as important players of the underlying economical community is an useful feature of the studied graph. The economical nature behind the graph arcs inspired us to define the concept of *active users*, defined as users holding a sufficient balance to be active part of the economic community, i.e. able to take part in transactions. We then studied the number of active users at passing time to have a better estimation of the real number of users in the Bitcoin community over time.

The concept of balance linked to nodes, allowed us to perform a set of analysis aimed to verify the *rich get richer* conjecture, studying the richness of nodes over time. We used the node connectivity as measure of richness as for traditional graphs, but we studied also the richness in terms of the balance held by the user. Our balances analysis results proved that the richness disparity increased over time while the richest nodes set remained more or less stable with passing of time, meaning that the wealth inequality in Bitcoin was increasing over time, concentrating more and more value in already rich entities. This study was performed by using several *ad hoc* measures, aimed at describing different aspects of concentration, and verified by using the Gini coefficient.

We do remark that our work was the first to perform such detailed analysis on an updated (at the time) version of the blockchain. All the related works in the literature had only attempted clustering older, and so smaller, versions of the blockchain, or avoided clustering at all, hence obtaining less significant results.

The set of performed analysis gave us a clear enough macroscopic knowledge of the user graph, so we concentrated our subsequent efforts in deriving some more precise information about precise users. In fact the oddities about the degree distributions and the diameter left us wondering what users were causing them. After a manual inspection of the transaction involved in both the indegree outliers and the high diameter we classified few peculiar transaction schemes. We then gave a general definition encompassing all the observed schemes, named *generic pseudo-spam transaction chain*. We then performed an automatic inspection of the transactions in the blockchain and studied their variables distributions. We not only proved that those transaction schemes, despite being a small percentage of the entire graph, were alone responsible for the unexpected macroscopic measures initially observed, but also connected such schemes with well known Bitcoin history events, such as the flooding attack of July 2015, and provided a possible explanation on their economical meaning.

We did not know of any other work performing automatic analysis of patterns at the time and we believe to have been the first ones to propose these kind of studies at the best of our knowledge. All the previous works on a similar topic were all limited to the manual inspection of unique case studies. The results gained are especially useful if used to strengthen the heuristic rules available in the literature, to obtain a more accurate addresses clustering. Our work in this field is still ongoing and we are currently working on the classification of other peculiar transaction patterns.

Blockchain Based Access Control Systems.

The second part of this thesis is dedicated to the contribution we have made in the field of Blockchain 3.0 applications. Our work has been focused on the integration of blockchain technology into Access Control systems. At the best of our knowledge we have been the first to propose such integration by also providing proof of concept implementations of each of our proposals. We have designed our proposed systems to take advantage of what we think can be the main contribution blockchain technology would give to a traditional Access Control system: transparency and auditability, i.e. users protection against fraudulent denial of access.

We have presented three main novel proposals on how to integrate blockchain within a traditional Access control system. The first proposal is a pluggable system to publish and manage Access Control policies on the blockchain. Thanks to the modularity of Access Control systems our proposed component can be plugged in and used by traditional systems without the need for further modification. By keeping the policy management process on the blockchain, the entire policy management history becomes transparent and so the corresponding access decisions becomes auditable.

The proposed component also introduces the novel concept of free *access right transfers*. This operation is introduced by us for the first time, and consists in the ability of the current access rights owner, i.e. the subject to whom the policy currently grants access, to transfer their right another subject. Furthermore the current rights owner can also split such rights and transfer them to different subjects. The right transfer is made through blockchain transactions, so it is transparent, not reversible and is carried out in a decentralised fashion without the need of for trusted intermediaries. Of course the underlying concept of value provided by an eventual cryptocurrency supported by the chosen blockchain can be used between subjects to pay such transfers. Depending on the application scenario, this novel operation can be an useful addition to a traditional Access control service.

For the proposed system we provide a reference implementation based on XACML policies and Bitcoin blockchain. To prove the proposal feasibility we deployed our system on the main Bitcoin network.

We then present two different novel proposals of Access control services complete overhauls. The Access Control services are in fact competently rewritten, shaped around the concept of *smart policy*, but keeping the same conceptual logic architecture. A smart policy is a self enforcing Access control policy implemented through a smart contract. This means that the policy management and decision evaluation are carried on by the blockchain in a decentralised fashion, and so beyond the control of the resource owner, to provide auditability. The other system components are implemented on the blockchain as smart contracts as well or not depending on the level of integration desired and the kind of resource the system is designed to protect. In fact we evaluate both an hybrid approach to control traditional resources, and a fully blockchain oriented approach to protect other smart contracts, named

smart resources. In particular this proposal is completely novel since no other proposals about smart contracts Access Control exists in the literature, to the best of our knowledge. Our completely novel contribution is also in the definition of Attribute Managers implemented as smart contracts and deployed on the blockchain as well, named *smart Attribute Managers*. In the thesis we also present a further proposal to plug smart Attribute Managers in a traditional system, to keep the blockchain integration at a minimum and without further modifying the rest of the traditional system, since attribute managers are already external pluggable services in traditional systems.

To prove the soundness and feasibility of our ideas we implemented a proof of concept implementation for each one of them, using the XACML standard and Solidity to write smart contracts to be deployed on the Ethereum blockchain. To evaluate the performances of our systems we presented a series of experimental results obtained by deploying our reference implementations on two different Ethereum testnets, including the Ropsten official one. The experimental results prove the meaningfulness of our novel proposals, proving how it can be already used in practice.

I.2 Outline

In the following, we describe the structure of the thesis and the list of the publications related to each part.

Part I: Bitcoin Blockchain Analysis :

Chapter I gives a brief introduction to the treated topics in the following chapters, as well as presenting a road map of this Part I.

Chapter 1 provides a basic background on digital currencies, and in particular cryptocurrencies. We use the Bitcoin protocol as use case explaining introducing the concepts of addresses, transactions, Bitcoin blockchain and distributed consensus. We then show some of the most studied attacks to Bitcoin available in the literature, including fraudulent mining techniques, double spending attempts and deanonymization attacks. Finally we present Ethereum, another popular cryptocurrency, the concept of smart contracts and Solidity, one of the widest adopted programming languages to implement smart contracts.

Publications

- Damiano Di Francesco Maesa. "Bitcoin Protocol Main Threats." Technical Report, Dipartimento di Informatica, TR . University of Pisa, Pisa, IT, 2017. `http://eprints.adm.unipi.it/2371/1/TechRep.pdf`

Chapter 2 presents our set of analysis performed on the Bitcoin users graph. We first explain the efficient clustering algorithm used and then perform a wide set of classical graph temporal analysis. We also study the rich get richer phenomenon both from a connectivity point of view and from an economical richness (i.e. users balance) point of view. Finally we briefly present the same set of analysis for increasing amount thresholds instead of time.

Publications

- Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Data-driven analysis of Bitcoin properties: exploiting the users graph. *International Journal of Data Science and Analytics (2017)*: 1-18.
- Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "Uncovering the bitcoin blockchain: an analysis of the full users graph." In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pp. 537-546. IEEE, 2016.
- Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "The Graph Structure of Bitcoin". Conference, under review.

Chapter 3 starts from the unexpected results found in the analysis performed in the previous chapter. We isolated the users and transactions responsible of such results and explain how a single transaction pattern adopted by a minority of transactions is responsible for both these two macroscopic effects on the graph. We try to give an explanation of why this peculiar transaction patterns are used by users, concluding that they are probably result of an artificial user behaviours.

Publications

- Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Detecting Artificial Behaviours in the Bitcoin Users Graph. *Online Social Networks and Media 3 (2017)*: 63-74.
- Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. "An analysis of the Bitcoin users graph: inferring unusual behaviours." In *Complex Networks and their Applications*, pp. 749-760. Springer, Cham, 2016.

Chapter 4 provides our concluding remarks about the work presented in this Part I. Furthermore we present our ongoing research to strengthen the user graph analysis, to define new heuristic rules and to deploy an efficient Bitcoin network listener.

Publications

- Damiano Di Francesco Maesa, Matteo Franceschi, Barbara Guidi, and Laura Ricci. BITKER: a P2P kernel client for Bitcoin. In *16th*

International Conference on High Performance Computing & Simulation (HPCS 2018). To appear.

Part II: Blockchain Based Access Control Systems :

Chapter II gives an introduction and a road map of this Part II.

Chapter 5 presents a sample of interesting Blockchain 3.0 applications. In particular we study end-to-end verifiable electronic voting, healthcare records management, identity management systems, decentralized notary, intellectual property protection and supply chains management.

Chapter 6 provides a brief background about Access Control systems. After defining what an Access Control system is and presenting its different proposed models, we present the Attribute Based Access Control model and XACML standard.

Chapter 7 shows our new approach based on blockchain technology to publish and manage Access Control policies. By leveraging a cryptocurrency based blockchain we also allow the distributed transfer of such right among users. We also provide a working reference implementation based on XACML policies, deployed on the Bitcoin blockchain.

Publications

- Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "Blockchain based access control." In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pp. 206-220. Springer, Cham, 2017.
- Damiano Di Francesco Maesa, Laura Ricci, and Paolo Mori. "distributed Access Control Through Blockchain Technology." *Blockchain Engineering (2017)*: 31, Ercim News.

Chapter 8 presents our proposal to codify Access Control policies as executable smart contracts on a blockchain, hence transforming the policy evaluation process into completely distributed smart contract executions. Not only the policies, but also the Attribute Managers required for their evaluation are implemented as smart contracts. We present two different reference examples depending on the type of resource we plan to protect. We introduce our reference implementation based on XACML policies and Solidity written smart contracts deployed on Ethereum. Finally, we evaluate the advantages and drawbacks of the proposal, evaluating the system performances through experimental results of our reference implementation.

Publications

- Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "Blockchain Based Access Control Services." In *IEEE Symposium on Recent Advances on Blockchain and its Applications, 2018 IEEE International Conference on Blockchain*, 2018, to appear.
- Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. "A blockchain based approach for the definition of auditable access control systems". Journal, under review.

Chapter 9 provides our conclusions about our proposals presented in Part II. After showing our current research directions, we introduce a possible extension to the blockchain based Access Control service showed in the previous chapter, by only implementing Attribute Managers as smart contracts.

Appendices :

Appendix A presents our BitKer kernel client employed as listener on the Bitcoin P2P communication network.

Publications

- Damiano Di Francesco Maesa, Matteo Franceschi, Barbara Guidi, and Laura Ricci. BITKER: a P2P kernel client for Bitcoin. In *16th International Conference on High Performance Computing & Simulation (HPCS 2018)*. To appear.

I

Bitcoin Blockchain Analysis

Introduction

In Part I of this thesis, we focus on the study of the Bitcoin *users graph* and its results implications on users anonymity.

We have seen in this thesis Introduction what a blockchain based distributed ledger actually is. In this part we start by giving a brief description of its main application, i.e. digital currencies, in Chapter 1. After an informal explanation of what a *digital currency* is and the social needs it satisfies we present the *Bitcoin* and *Ethereum* protocols in Section 1.2 and Section 1.6 respectively. We chose to focus on these two cryptocurrencies only both because they are arguably the most used and known digital currencies nowadays and because our work presented in the rest of this thesis is based on them. We present Bitcoin in more technical depth to allow the reader to familiarize with cryptocurrencies though the first and most famous example. We show the core rules of Bitcoin protocol (showing what addresses and transactions are and how they are used in a distributed consensus protocol to build the blockchain) and we also present three main attacks to the protocol to highlight its limitations. In particular, we show the main fraudulent mining techniques employable by fraudulent miners to unfairly gain an advantage in the mining race (Section 1.3), the attacks carrying double spending attempts (Section 1.4) and the threats to users privacy in Bitcoin (Section 1.5). The subsequent presentation of the Ethereum protocol mainly highlights its differences with Bitcoin.

In Chapter 2, we present how we obtained the *transactions graph* from the blockchain and the *users graph* from it through the most widely accepted heuristic rules (the *multi inputs heuristic* shown in Section 1.5.2). To better understand the graph and draw further insight from the blockchain we performed a set of analysis on the users graph, presented in the same chapter. More in depth, we present our support for the analyses of huge amount of data and describe a set of analysis of the Bitcoin users graph performed by that support. The analysis spots peculiar topological properties of the users graph, which are new if compared with other complex networks. The topological observations on the users graph can translate in emerging economical trends. In this sense, we can highlight economical outliers through the observation of topological phenomena and verify economical hypotheses, as rich get richer.

In Chapter 3 we use the useful insight gained from the analysis of Chapter 2 to detect and model peculiar artificial behaviours in the blockchain. We present a set of analyses revealing the presence of a set of unusual topological patterns in the Bitcoin users graph, for instance we detect set of outliers in the indegree distribution of the graph. We show that these patterns are not due to normal economic behaviours, but to artificial transactions whose nature is conjectured in the text. Aim of this work

is to isolate distinctive user behaviour patterns, not only to better understand users interactions but also to design better heuristic rules to strengthen a deanonymization attack.

Finally in Chapter 4, we present our future work plans and current research, including our undergoing work about the *external information gathering* step of the attack.

One of the leading threads of the chapter is the concept of user privacy in the Bitcoin protocol. We explain in Section 1.2.1 how the Bitcoin protocol does not provide strong anonymity, i.e. user privacy is only weakly protected through *pseudonymity*. We then present in Section 1.5.2 the state of the art attacks on users privacy in Bitcoin. The scheme of the attack is shown in Figure 1.10 and the three main steps are *blockchain scanning*, *heuristic based clustering* and *external information gathering*. In Chapter 2 we perform the *blockchain scanning* and *heuristic based clustering* steps and show macroscopic properties about users that this allows us to infer. We use the insight gathered to refine our analysis in Chapter 3 to investigate some peculiar transaction patterns. The information gained help assess the privacy level really offered to users as summed up in Chapter 4, where our ongoing research is shown.

1

Cryptocurrencies

Abstract

In this chapter we provide a basic background on digital currencies, and in particular cryptocurrencies. We use the Bitcoin protocol as use case explaining some of the protocol technical details, showing how addresses are used into transactions to define the state of the system, and how transactions are recorded into a blockchain to remember such state. We introduce the problem of keeping the state consistent among the participants and explain how the distributed consensus is used to solve it and to incentive users to behave honestly. Furthermore, we show some of the most studied attacks to Bitcoin available in the literature, including fraudulent mining techniques, the most successful malicious double spending attacks, and anonymity breaking attacks. Finally we present Ethereum, another popular cryptocurrency, the concept of smart contracts and Solidity, one of the widest adopted programming languages to implement smart contracts.

We present cryptocurrencies by first introducing the concepts of *digital currency* and *cryptocurrency* in Section 1.1. We explain why they were introduced, how are they different from traditional forms of currency, what properties they need to guarantee, what new challenges they face and what is their history.

To give a practical example of a cryptocurrency based on blockchain technology we present in Section 1.2 the Bitcoin protocol, explaining in depth how the Bitcoin protocol works. We first give an overview of the main idea behind it and present a brief history that led to it. We then present the basic component of the protocol, *addresses*, in Section 1.2.1. We explain how they work and how they protect users identities, including observations about post-quantum addresses security. In Section 1.2.2 we present *transactions*, and how they can be used to move funds between addresses. We explain how transactions are recorded in a decentralized database, called the blockchain in Section 1.2.3. We show how Bitcoin solves the problem of reaching a distributed consensus between untrusted entities on the state of such database. To do so we first explain what is a Proof of Work and how it is used in Bitcoin. We then explain how the Nakamoto consensus adopted by Bitcoin can solve forks and present some proposed alternatives to Bitcoin Proof of Work.

The subsequent three sections are dedicated to the study of some of the Bitcoin protocol main shortcomings and menaces by showing some of the most studied attacks available in the literature.

In particular in Section 1.3 we present the state of the art fraudulent mining techniques. The attacks shown in Section 1.3.1 prove how the incentives provided by the protocol to encourage miners to behave honestly might not be enough in the presence of other real world gains. In Section 1.3.2 we show the selfish mine strategy and other similar block discarding attacks to prove how fraudulent miners can employ techniques to obtain a relative profit greater than the honestly expected one. This effectively proves how mining is not fair, i.e. miners do not necessary gain a relative expected reward proportional only to the computational power deployed. Finally in Section 1.3.3 we explain why miners have incentives to join mining pools and how fraudulent miners can exploit them.

In Section 1.4 we show the state of the art attacks attempting a double spending. To dos o we explain the importance of transaction confirmations wait. We then show why waiting for confirmations is not always practical and it is desirable to use *fast payments* instead. We explain two different kinds of attacks against fast payments depending if the attacker is also a miner or not.

Finally, in Section 1.5 we explain the level of protection of users privacy offered by Bitcoin. We present Network listening techniques in Section 1.5.1, highlighting its dangers for users anonymity. We present these techniques application to Bitcoin communication network topology discovery and malleability attempts monitoring. We present the state of the art about the mentioned techniques. In Section 1.5.2 we show the Bitcoin blockchain deanonymization attack, showing the attack scheme, the main heuristic rules used, its possible benign results and the proposed countermeasures to restore users privacy. Finally in Section 1.5.3 we explain the Simplified Payment Verification used by thin nodes and the possible dangers it poses to users privacy.

We conclude the chapter by presenting another cryptocurrency protocol, Ethereum, in Section 1.6. We first show its similarity with the Bitcoin protocol explained in Section 1.2, especially concerning the underlying currency and distributed consensus used. We then explain the use of smart contracts run by the Ethereum Virtual Machine and the concept of gas. Finally, in Section 1.6.1 we provide an overview of Solidity, one of the programming languages used to write smart contracts to compiled and run by the Ethereum Virtual Machine.

1.1 Digital Currencies

Internet changed our society starting the so called “digital revolution”. Almost every individual can now directly connect to any other on the globe allowing for a point-to-point information exchange or, particularly, commercial interaction. But still, for most users, a direct point-to-point value exchange (payment) is not possible,

and a third party financial intermediary is required. Entrusting a third party with their funds weakens the user's control and spending freedom, limiting them to have commercial interactions only with parties recognized by the entity. An example of this was the 2010 Wikileaks blockade [274], when users were prevented by their financial intermediaries (Bank of America, VISA, MasterCard, PayPal and Western Union) to spend their own funds as they pleased. This episode confirmed that centralized currencies controlled on our behalf by third party financial organizations are not the best way to freely and anonymously exchange value between users in a global distributed community. This need in the new digital society is met by digital currencies.

The Concept of Currency. The only purpose of a currency is to hold value and allow the exchange of that value between users. Originally, currencies were born to overcome the limits imposed by barter. Firstly, precious commodities (mostly gold) were used to physically represent the value, then we moved to certificates backed by some precious commodities assets, to finally arrive to fiat currencies which are certificates without any backing. Gold was used to directly represent the value owned, gold certificates were backed by gold to indirectly represent the same value. The certificate value is indirect because it is not determined by the value of the paper and ink it is printed on, but by the amount of gold the certificate represents. Its acceptance is based on the trust of users in both the emitting entity and the intrinsic value of gold it represents. A fiat certificate is not backed by any precious commodity value, so its acceptance is only based on users trust in the emitting entity. This means that we accept fiat currencies just because we expect every one else to accept them as well. So we socially agree they represent some value. This is similar to what happened with gold thousands of years ago, when people where willing to trade useful things (like food and clothes) for a mostly useless but shiny metal.

Digital currencies make the difference between abstract value (the value the certificate represents) and material value (the value of the material support the certificate is printed/minted on) of a certificate explicit. Considering that the certificate abstract and material values are not correlated, we can as well eliminate the material support obtaining just pure value representation. The value represented is now a free information electronically remembered. This is the same step performed by "electronic money" where value represented by traditional fiat currencies emitted by central entities (central banks) is electronically stored by third parties. Electronic money is not a form of digital currency, because it is just traditional currency digitally processed. That is why a digital currency requires a further step. The currency should be distributed, and exchanges between users should be direct, without the need of third party or other users intervention.

Novel Operations. To be distributed, a digital currency needs to decentralize two operations to the entire community:

- the minting of new coins;

- the validation of transactions exchanging value.

The problems of performing those two operations are usually solved with cryptography, that is why some digital currencies are also called "cryptocurrencies". The whole idea is to shift the users' trust from a human controlled central entity to few reliable cryptography functions.

It is usually said that with digital currencies every user is their own bank [17]. That is because each user is the only one in charge of controlling and managing their own funds, they can check the validity of every other user transaction and can partake in the minting process. Any user can freely join or leave the community without costs and cannot be banned by others. Only the rightful owner can access their own funds and so no funds can be frozen or seized. Usually, digital currencies are also compared to electronic cash, but to behave like cash additional properties must be fulfilled:

- non-traceability
- non-reversibility
- direct exchange

Among those properties we need to point out that non-traceability and direct exchange together give anonymity.

Double Spending Prevention. As we have already explained, designing digital currencies introduces new problems unknown in traditional currencies design. The most notable problem is called "Double spending". Preventing double spending means preventing users from spending the same funds in more than one valid transaction. This problem does not exist in a traditional setting because each user account is controlled by a centralized entity (bank) who has full control over user's funds. It is the central entity the one tasked of issuing transactions, not the user (as happens with digital currencies), and so there cannot be double spending attempts by the users (at least not without entity errors). In a distributed scenario, each user is tasked with creating their own transactions, and so we need to define a method that allows users to reach a consensus on which transaction is to be declared valid in presence of multiple conflicting ones (to prevent double spending). Bitcoin double spending solution is to allow every user to know all the previous validated transactions of all users. The entire history is needed because a user cannot recognize a transaction as a double spending attempt if they do not know the original transaction[1]. That means that the entire history of all transactions ever validated is publicly visible to everyone. This leads to a clear threat to users privacy if the identities of users involved in transactions are not protected enough (see Section 1.5).

[1]The transaction whose funds are attempted to be spent for the second time.

1.2 Bitcoin Protocol

Bitcoin History. The idea of digital currencies is not new: the 1982 paper "Blind signatures for untraceable payments" by David Chaum [55] can be considered their manifesto. Chaum was the first to propose the use of cryptography to validate transactions rather than protect them. He also founded a commercial society called DigiCash in 1990 to promote this new cryptographically secured currency but still based on a centralized entity.

Bitcoin is the first cryptocurrency (expressed as BTC) to have reached worldwide popularity and adoption but it is not the first digital currency and its core ideas were already available ten years before its announcement. As we said in the previous paragraph, the first cryptocurrency attempt is considered to be [55] by David Chaum, where the author introduces blind signatures to "allow realization of untraceable payments systems which offer improved auditability and control compared to current systems, while at the same time offering increased personal privacy" [55]. In 1997 the cypherpunk cryptographer Adam Back invented the first proof-of-work called hashcash [10, 9] as a DoS countermeasure applied to e-mail spamming. We will explain the concept of proof-of-work in details in Section 1.2.3, for now it suffices to informally say that it is a method to define a computationally hard task that has to be executed before accessing a resource or validating a procedure. In 1998 Wei Dai released b-money [69], an anonymous digital currency employing proof-of-work, and at the same time Nick Szabo invented bit-gold [253], which used proof-of-work schemes to simulate gold digging hardness and had a controlled inflation rate.

Bitcoin Community. Bitcoin is a cryptocurrency and that means that it cannot be seen just as a distributed protocol. The Bitcoin real world community influences its development as much as its original design principles. Hereafter we will concentrate on the technical topics of the protocol, but studies are also available on the more social aspects of Bitcoin. Some papers have studied the Bitcoin community and ecosystem like [229, 135], and [59] where the real world use of Bitcoin in the Silk Road dark web marketplace is presented. Even the FBI has produced its own record about Bitcoin [111], showing concerns about its use as payment for illegal activities. More works have been done studying Bitcoin economy [92, 139, 278] including financial groups assessments for speculative purposes [275, 199, 236]. Works has also been done to try to explain the Bitcoin exchange rate, like in [171] where a correlation is found between the first 2013 price spike and Cyprus bailout. Today the biggest obstacle for serious business ventures in entering the Bitcoin economy has been the lack of a clear legal definition of cryptocurrencies. Some guidelines have been released by international organizations [104, 202, 14, 144] but still every jurisdiction applies different legal frameworks and tax rules to digital currencies [97, 278, 250, 174, 16].

Protocol Overview. Users take part in the Bitcoin economy through addresses (see Section 1.2.1). An address is a double hash of a public key derived from a

ECDSA key pair. The address (and hence the public key) will be used by the user to send and receive payments, while the private key will be used by the user to provide proofs of ownership. Pseudonymity, i.e. lack of linking between addresses and identities or other addresses, is the only (weak) anonymity protection in Bitcoin.

To exchange funds between addresses, multi input (address from which funds are withdrawn), multi output (address where funds are stored) transactions are created (see Section 1.2.2). Transactions are the only mean to manage funds, so funds can be divided or aggregated only by being spent. Funds are represented by a transaction chain showing the passage of value (split and merge) between addresses, validated at each step by the previous owner signature. New transactions are created by any user and notified to the community with a gossip style broadcast message on the P2P Bitcoin network. Furthermore, a special kind of transaction called *coinbase* exists to allow for new value creation (distributed minting of new coins as part of the validation process) and fees collection.

Bitcoin solution to the *double spending problem*, i.e. the same funds spent more than once by different transactions, is to remember the history of all the past transactions in a so called blockchain (see Section 1.2.3). Transactions are grouped in blocks linked in a chain and the linking between blocks is achieved by saving the hash of the header of the previous block in the next block header. It is necessary to reach a distributed consensus to choose which block (and so which transactions) to add to the chain, because there could be incompatible transactions caused by a double spending attempt. The distributed consensus protocol introduced and used by Bitcoin is called Nakamoto consensus and relies on HashCash Proof-of-Works. Any user can choose to take part in the consensus protocol and become a validator. Validators are called *miners* and the entire validation process is called *mining*. The consensus protocol is divided in steps. At each step every participant chooses a list of valid transactions and builds a block out of those. Then a POW based distributed process is employed to choose at random one of the participants to publish his block to be added as next block on the head of the blockchain. Each participant then chooses if the newly published block is to be accepted or rejected. This choice is manifested by deciding to start looking for the next block on top of the new one (accept)or by keeping looking for a block on top of the old chain, ignoring the new block (reject). In the latter case we say that the blockchain is experiencing a *fork*, since, during the next step the participants will look for new blocks on different branches of the chain. In case of conflicting chains coexisting at any given time the protocol dictates to look for new blocks (and add them) only on the longest chain branch, so that the shortest branches will eventually fall behind and be ignored.

In the following we will explain the basics of Bitcoin protocol. Do note that there is no "official" protocol specification available for Bitcoin, so we refer only to the main well defined basic topics of the protocol as originally presented in [195], updated when necessary with the information available from the official Bitcoin wiki [27], Bitcoin improvement proposals [21] and official client implementation [25].

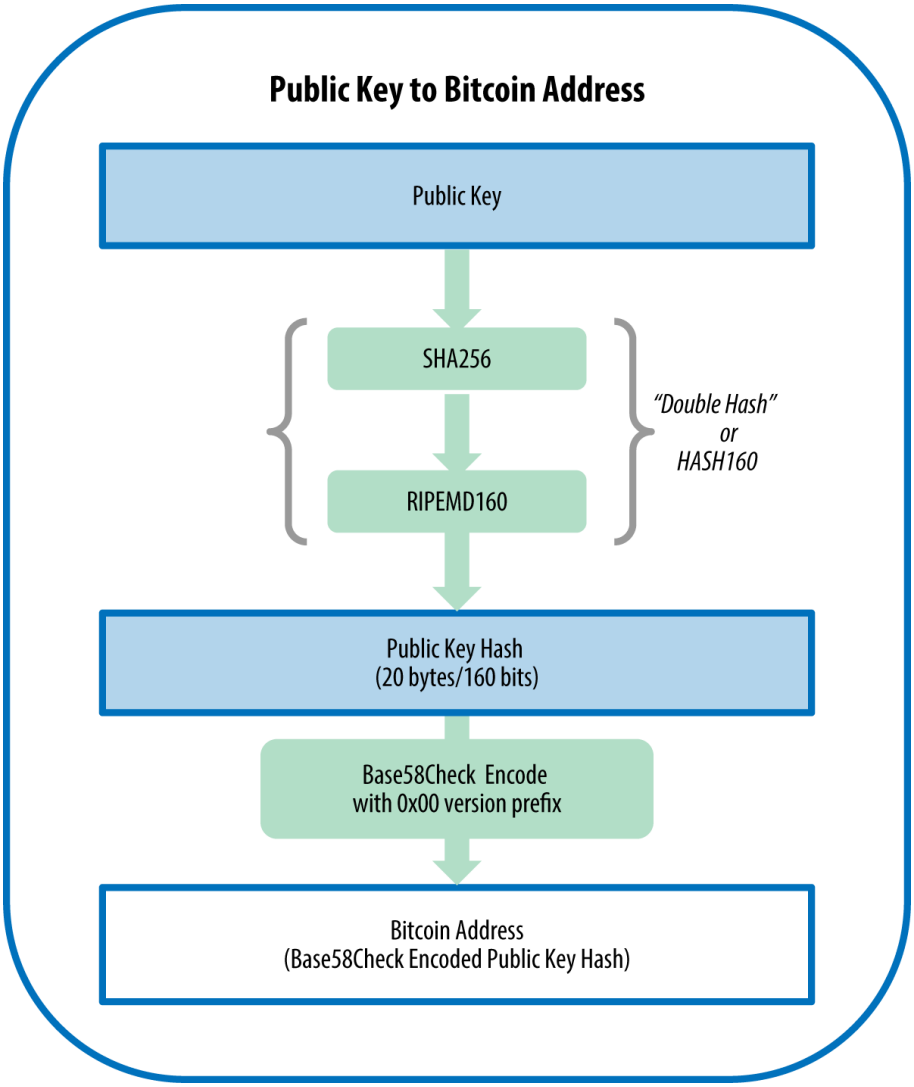


Figure 1.1: Bitcoin address construction. Source [8].

1.2.1 Addresses

Address Definition. To enter the Bitcoin economy, each user only needs to generate a new asymmetric cryptography key pair. The public key is used to obtain an address to send and receive payments, while the private key is needed to prove that the user is the true owner of that address. The private key is used as proof of ownership, because each transaction spending funds contained in the corresponding public key (address) must be signed with the correct private key to be accepted. That is the only ownership proof available, so ownership of an address just means knowing the corresponding private key.

The public key cryptography used by Bitcoin is ECDSA [37], so the private key is a random 256 bits long number and the corresponding public key is 512 bits

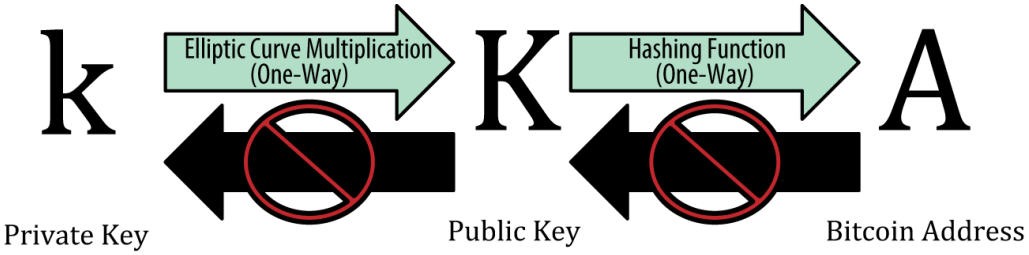


Figure 1.2: Information flow in a Bitcoin address. Source [8].

long. But each user does not advertise their public key as it is. They firstly hash it with SHA-256 [198] to obtain 256 bits on which they apply RIPEM-160 [214, 89] to obtain 160 bits. Those bits are encoded with the Base58Check encoding to obtain an ASCII representation which is used as the users public address. The Base58Check encoding is a purpose built encoding containing a check-sum (the first four bytes of the double SHA-256 hash of the public address) and avoiding the use of similar characters symbols. The public address can be advertised and must be known to all those who want to send payments to it.

Address Security. We observe now that it is not possible to learn the public key of a user from an address because the hash functions used are cryptographic and so irreversible. So the address is used in place of the public key to increase the secret key security, because it restricts any attacker who tries to recover the private key from the public key to a limited time window. The public key cannot be kept secret forever, because it is necessary to verify a signature (necessary to make a payment). That is why the public key is attached to any transaction spending funds from the corresponding address, so that an attacker has to wait for this transaction in order to learn the public key. The private key is necessary to spend the funds contained in the corresponding address, but the attacker can try to recover it only after a transaction spending those same funds is revealed. This means that the attack can only be attempted during the time that passes from when the transaction is announced to when it is validated (and so the funds are spent and cannot be stolen anymore). The security improvement of this design can be better appreciated considering the example of post-quantum cryptanalysis [20].

Post-Quantum Scenario. Up to date one of the best known quantum algorithm to break cryptographic hash functions is the Grover search algorithm [20], which gives a quadratic speedup over classical algorithms. So this algorithm allows to halve difficulty of hash functions and symmetric cryptography breaking in a quantum scenario. A cryptographic hash function requires 2^n computations to be inverted and $2^{n/2}$ computations to find a collision on classical computers, so with a quantum computer we will need $2^{n/2}$ and $2^{n/4}$ computations respectively, so the problem remains intractable. The Shor quantum algorithm [218], instead, allows to solve the discrete logarithm problem in polynomial time. ECDSA security is based on

this problem intractability, so public key elliptic cryptography would not be secure anymore.

As we have seen before, the public key is revealed only when a payment is issued, while to receive payments only the address needs to be known. Hash security is reduced but not compromised by quantum computers (it would be enough to simply increase the size), so a quantum attacker would not be able to learn the public key from the address. But once they know the public key they would be able to find the correct private key in polynomial time. So if public key were used instead of addresses the protocol would be compromised. Using addresses allows to restrict the quantum attacker to a limited time frame (dependent on validation time) to try to find the private key. If the validation time is small enough the attack would be fruitless because the attacker would always be looking for private keys protecting funds already spent. So the only effect of a quantum computer would be to force users to use each address (and so each key pair) only once. We note that even if the validation time is significant enough to allow the attacker to find the secret key in time they would anyway be at a disadvantage. Knowing the private key, the attacker could create a different valid transaction sending the victim funds to an attacker controlled address. But this transaction will be in conflict with the victim's original transaction and so only one of the two transactions would be validated. Following the protocol rules, the first transaction to be relayed on the network is the one with higher probability of been accepted (the exact probability depends on the percentage of validator nodes reached before the other transaction). The time spent by the attacker to find the private key would result in a propagation advantage of the honest transaction, which will then have a higher chance of being accepted than the fraudulent one.

The attacker could improve their chances by combining the private key discovery with a sybil attack, to isolate the victim from honest nodes, providing an alternative reality to the victim. The attacker would listen to the victim's transactions without forwarding them to honest peers, and forwarding their fraudulent transaction instead, once it is ready. More generally, relying on a broken public key algorithm would be pointless *per se*, so it would be necessary anyway to use a quantum resistant asymmetric cryptographic scheme (like lattice cryptography).

Address Anonymity. A side effect of ECDSA security is that if a private key is lost it cannot be recovered. So any funds stored or sent in the future to the corresponding address would be lost forever. We also note that creating new ECDSA pairs (and so addresses) is not expensive at all and so each user can create and use different addresses. This leads to the use of pseudonyms, which means that each address is a user alias without any kind of information linking it to the user or other addresses created by the same user (see Figure 1.3). Pseudonymity is the only (weak) anonymity protection in Bitcoin. To improve transactions anonymity it is recommended to create a new address for every transaction. This is not computationally expensive but it can lead to an address management problem, if the

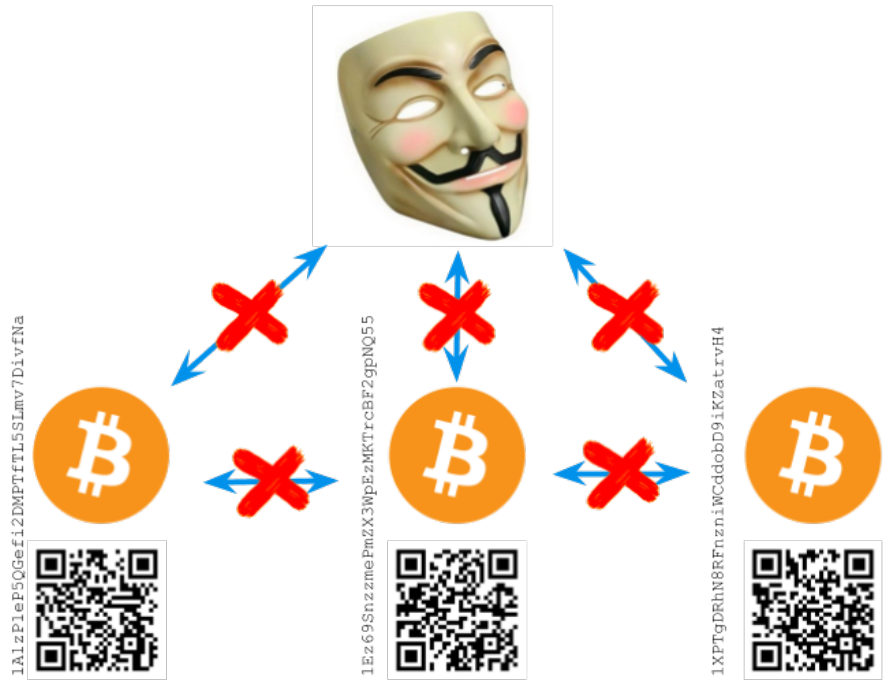


Figure 1.3: Bitcoin pseudonymity property. There is no information linking neither between an address and the controlling entity nor between different addresses controlled by the same entity.

number of addresses keeps increasing. For users with high daily transaction volumes (such as exchanges and merchants) issuing a new address each time would lead to an overwhelming amount of addresses. To simply discard empty old addresses could be dangerous, because someone could still send payments to them, resulting in forever lost Bitcoins.

1.2.2 Transactions

Transaction Definition. A transaction is a funds exchange between addresses. Transactions are multi input, multi-output, i.e. a transaction may have more than one input (address from which funds are withdrawn) and more than one output (address in which funds are stored). Each transaction completely transfers funds from the inputs to the outputs (no change is left in the input addresses). So to keep the change (difference between input sums and the sum we actually want to pay including fees) a user should include among the outputs an address they own. As we already stated, transactions are the only mean to manage funds, so funds can be divided or aggregated only by being spent. That is possible because a transaction involves addresses and not users and every user can have different addresses, so a user can use a transaction to split, merge or move funds between their own addresses.

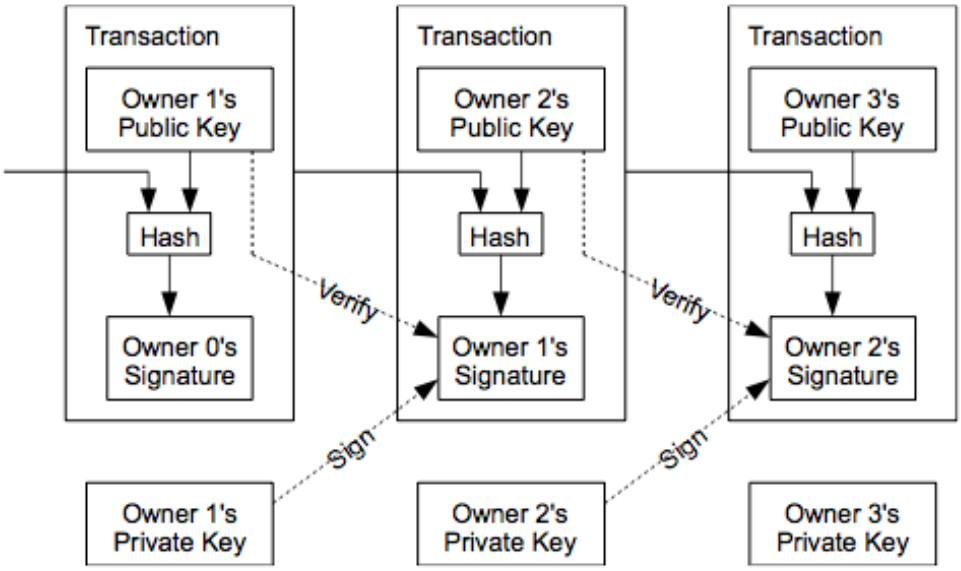


Figure 1.4: Chain of transactions (single input, single output for simplicity), showing the use of digital signatures to prove funds ownership. Source [195].

Transaction Fees. If the sum of input values exceeds the sum of output values[2], then the exceeding value is considered a voluntary fee paid to the validator to cover the validation process expenses. Paying a fee is optional but it is considered fair practice and it can shorten the validation time of the transaction.

Transaction Chains. In a transaction each output can be seen as a couple (Bitcoin amount, receiver address). On the contrary, each input specifies where to withdraw the funds, so it does not indicate an address but instead the previous transaction (actually its hash) where the funds were stored in the address now used as an input (see Figure 1.4). Funds are represented by a transaction chain showing the passage of those funds (split and merged) between addresses, validated at each step by the previous owner signature (see Figure 1.5). In Bitcoin, transactions alone specify the entire state of the system. There is no "coin" exchanged between users, the coins are implicitly represented by the flow of value through transactions. We can informally state that a "coin" in Bitcoin is its own "history".

Transaction Scripts. Although from a high level we can say that transactions protect funds ownership through digital signatures, in practice this is achieved by more complex scripts. The Bitcoin protocol uses a non-Turing-complete stack based scripting language, and scripts are (mostly) used in a transaction to specify conditions needed to redeem the funds of that transaction. The most common example of such condition is a signature, but it is not the only possibility and scripts can be

[2]If the reverse happens the transaction is obviously invalid.

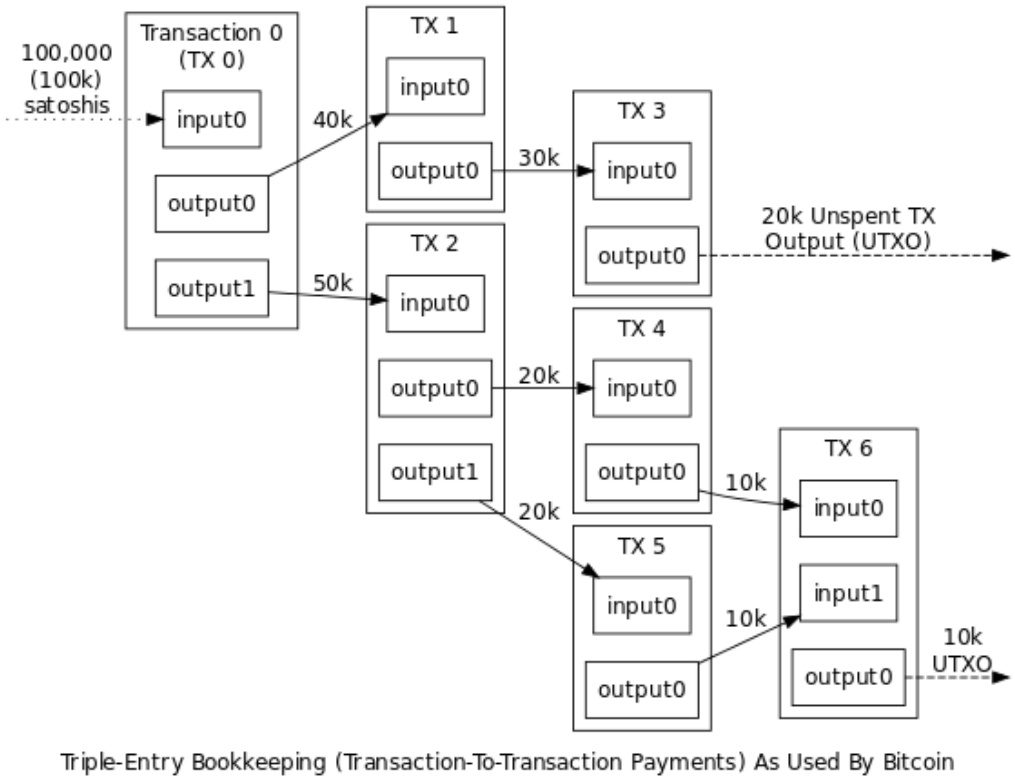


Figure 1.5: Example of how transactions can be chained together. Source [26].

arbitrarily complex. When a transaction is tested for validity, the input scripts are concatenated with the output scripts, evaluated, and all transaction scripts must evaluate to true for the transaction to be validated. In practice only few types of standardized scripts are used in transactions, they, and the transactions using them, are called *standard*. What's more important is that non-*standard* transactions (so transactions containing non-*standard* scripts) are accepted but not relayed by compliant nodes, so they have less chances of actually ending up in the blockchain. The most used *standard* script types are called Pay_to_PubKey_Hash (`p2pkh`), Pay_to_PubKey (`p2pk`), Pay_to_Script_Hash (`p2sh`) and Pay_to_Multisig (`p2ms`).

1.2.3 Consensus

The digital signature guaranties that only the rightful owner can spend their funds, but the so called "double spending problem" is that it does not prevent them from spending the funds more than once in different transactions. Bitcoin solution is to remember the history of all the past transactions to determine the actual owner of every fund at each given time. In case of conflict between different pending transactions it is necessary to reach a distributed consensus on which transaction to add to the official history. The history is maintained in a distributed database

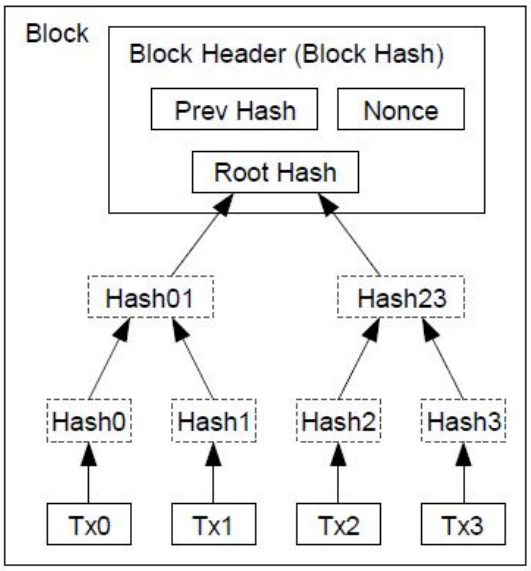


Figure 1.6: Use of merkle trees to make the header dependent of all transactions contained in a block by only storing the root hash of the tree. Source [195].

of all the valid transactions ever happened, which is a distributed timestamping service because it fixes transactions in discrete time instants, allowing for a partial chronological ordering. The database is called blockchain because transactions are grouped in blocks linked in a chain[3]. The linking between blocks is achieved by saving the hash of the header of the previous block in the next block header. To make each block header (and so its hash) dependent from all transactions contained in that block, the root of the (implicit) merkle tree [182] (i.e. a binary tree of hashes where data is contained only on the leaves and each node contains the hash of the concatenation of the two children) built from the block transactions hashes is included in the header (see Figure 1.6).

Proof of Work

It is necessary to reach a consensus to choose which block (and so which transactions) to add to the chain, because there could be incompatible transactions caused by a double spending attempt. The idea is to let every user (who wants) vote using their computational power. It was chosen the "one CPU one vote" model instead of the "one user one vote" to protect the protocol from fake identities, vulnerable in an anonymous P2P network naturally prone to sibyl attacks. In fact, Creating new users is cheap, while dedicating computational resources is not. So the "one CPU one vote" strategy gives proof of a true commitment. A user participating in the validation process is called miner and the process itself is called mining. Bitcoin uses

[3]It is actually a tree because of chain forks, as we will see later.

Proof-of-Work (PoW) [95] to implement the voting process through computational power.

PoW Definition. Finding a proof-of-work means finding the solution of a computationally intense cryptographic puzzle to prove that some amount of effort was spent. The puzzle used should be asymmetric, which means that it should be computationally difficult to find a solution, but, given a solution, should be computationally easy to verify it. It should also allow to adjust the difficulty (of finding the solution, not of verifying it), and it should be dependent from some parameter to allow having different puzzles with the same difficulty. This is needed to make the solution effort independent of any *a priori* computation.

Bitcoin PoW. In Bitcoin, validating a block means finding a hashcash [10] type proof-of-work with double SHA-256 of the block header. It means finding a Nonce value so that the double SHA-256 of the block header is less that an established Target value. The Target value is automatically updated every 2016 blocks (approximately every two weeks) considering the computational power of the entire network (estimated by the average time passed to validate a block) in order to keep the average validation time of a new block around ten minutes.

Solving this type of proof-of-work is equal to a hash partial inversion, but, because the hash function chosen is cryptographically secure, the best known method is a brute force attack. This means that the best method is trying different random nonces until one satisfies the PoW, so the average time spent depends only on the computational power (more precisely the hash power) used. This cryptographic puzzle is a PoW because it is computationally expensive to solve but constant to verify (requires only one hash computation), the difficulty is adjustable (changing the target value) and the puzzle is parametric, the parameter being the block header, so it is not possible to work on the puzzle without knowing the block (no a priori advantage is possible).

Bitcoin PoW is a probabilistic PoW, that means that finding a solution is computationally expensive on expectation. The randomization is important because otherwise the miner with the biggest computational power will always find the solution first and so would be the only one to produce blocks. The randomized PoW instead allows miners to have a probability to find each block proportional to the hash power dedicated [4].

PoW Rewards. To encourage mining and repay miners for their computational expenses, to each new block is associated a reward collected by the block finder. The reward consists in the sum of all fees of the transactions contained in the block, plus a fixed amount of new coins. The fixed reward starts from 50 BTC and halves over time every 210 000 blocks until it will became zero at the 6 930 000th block, which is expected to be mined during the year 2140. The decreasing coin minting is adopted to reflect a deflationary money supply growth and also solves the distributed

[4] This is true as long as no single miner controls more than half of the total combined hashing power.

minting problem. This reward is credited to the miner, allowing it to add a special transaction, called *coinbase*, to their newly found blocks. This transaction has no inputs but only output addresses to whom the sum of newly minted value and all fees from transactions in that block are credited.

Distributed Consensus

Reaching consensus in a distributed system (that can be as simple as reaching a common decision over a value) in the presence of byzantine faulty entities is a well studied problem. By byzantine faulty entities we mean we do not make any assumption on the kind of failure the entities can incur in. This allows us to cover any possible kind of situation and attacker, for example both random crashes and active malicious behavior. In [115] it was proven that for deterministic asynchronous message networks cannot exist a reliable consensus protocol resilient in the face of even a single entity failure.

In [186] it was proven that Bitcoin does not achieve consensus only with negligible probability when the adversary controls less than half the overall computing power, under the model of anonymous synchronous network. It was also shown that the protocol is scalable, in that the running time and total message bits are all independent on the size of the network, instead depending only on the ratio of faulty processes.

Note that the actual Bitcoin network is not synchronous, but as was shown in [119] it is enough that the network synchronizes much faster than the PoW solution rate. It means that the assumption holds true as long as the messages propagation time (currently in the order of seconds [73]) is much smaller than the expected time to solve a PoW (ten minutes on expectation by design). In [119] similar results as [186] are shown, showing a proof that a majority of honest miners will agree on a common prefix of the blockchain, growing in length over time. Even in the light of those formal results we should always remember that the use of monetary rewards introduces new problems in the study of consensus, because this kind of incentive can be seen differently by different entities, and so it is difficult to formally define it.

Other Kinds of Consensus

Bitcoin SHA-256 PoW is not the only possible consensus mechanism used in cryptocurrencies. Many altcoins derived from Bitcoin changed the algorithm used or the concept of PoW itself.

ASIC-Resistance. Currently, mining is performed on ASICS devices, built ad-hoc to have performances and power consumption orders of magnitude better than CPUs (and GPUs). This has driven out occasional miners, and has transformed mining in a costly business with a high entrance cost, menacing decentralization. So there have been proposals to use PoW algorithms difficult to implement on ASICS devices. The

ASIC-resistant algorithms proposed so far are memory intensive, which means that they require efficient access to big memories, feature lacking in traditional ASICS devices. The most famous one is "scrypt" [207, 208, 209] used by the most famous altcoin Litecoin [169]. Unfortunately, ASIC devices built for scrypt are already available. Also, the CPU/GPU mining inefficiency has botnet [280] protection as a useful side effect. In [137] a survey of the Bitcoin botnet ecosystem of the time is presented. The paper shows how most of the botnets were directly participating in well known honest mining pools and how the increasing difficulty has reduced their profitability.

Useful PoWs. Another proposal was to try to have a PoW whose solution is also a useful result as a side effect [153, 215, 216]. The issue is finding useful problems which are characterizable as PoWs, meaning they should allow for proportional difficulty adjustment and no information should be pre-computable.

Beyond PoW. There have also been proposals to completely substitute the PoW scheme. The most famous proposal is called Proof-of-Stake (PoS) [206, 154, 136] and consists in proving to be the owner a certain amount of currency. The idea behind this is proving to be involved in the currency. This is mostly done by consuming (i.e. resetting to zero) coin age. The coin age of a sum of coins is defined as the product of the amount of coins and the time since last transaction involving that sum. So coin age is directly proportional to the amount of coins and to the quantity of time during which those coins were not used. Then, a special transaction (containing a self reward) is created from a user to themselves spending those funds and so resetting their coin age. This transaction is a valid PoS (and may be used to validate a block) with probability inversely proportional to its coin age. The PoS is computed on only static data except for the timestamp dependent coin age, so it is independent from the computational power owned.

The main advantage of this approach is that it would be more difficult for an attacker to gain control of large sums rather than to buy specialized hardware. It would also be more expensive for them to perform disruptive attacks to the currency, because they would own a lot of that currency themselves and so they would be the first to be damaged by it.

Other proposals include Proof-of-burn (to prove its own commitment a user creates a transaction destroying some coins by sending them to a non redeemable address) and Proof-of-retrievability (mining requires solving a classical PoW but using an algorithm that requires to locally store some data from a public dataset. This allows mining to have a public useful side effect)

Forks

Bitcoin is proven to reach an eventual consensus (under the assumptions previously stated). It means that the network will agree on an ever-growing common prefix of the blockchain, but the most recent blocks could be different. When an honest

miner finds a new block they announces the the new block immediately but, due to message propagation latency, another miner, currently unaware of the new block, could find a different new valid block themselves. So two different but both valid blocks may coexist. In this case we say a "blockchain fork" has occurred. Each miner then votes on which block to deem valid by starting to look for the next block to add after the chosen one. It means that the miner's vote consists in dedicating their computational power to increase the branch they deem valid. The protocol rule imposes to consider valid only the longest branch, where "longest" means the one with cumulative higher difficulty, and hence more computational power (so more votes) dedicated to its creation. Eventually one branch will grow longer than the other and every honest miner will start looking for new blocks only on top of that branch, *de facto* pruning the loosing one. So a temporary consensus is reached. The transactions in pruned branch and not included in the other one are still considered not confirmed, and they will have to wait to be include in a new block again.

1.3 Fraudulent Mining

In the previous section we have explained how Bitcoin needs the majority of the computational power to be honest to ensure that a consensus is reached. The original idea [195] was that the reward system would have been an incentive for miners to remain honest. This, however, does not hold true in the face of two problems:

- It assumes that the Bitcoins awarded are the only value precious to users, while in reality they alone might not reflect the entire real-world profit.
- It assumes that miners will gain an expected reward that is proportional to the percentage of computational power controlled, no matter how the miners behave. That is not true in the face of fraudulent mining techniques such as block discarding attacks.

1.3.1 Real World Gains

Miners are not just interested in the number of Bitcoins they can gain, but rather in the real-value of those Bitcoins. If a miner was to use known fraudulent techniques to gain more Bitcoins while at the same time undermining the public trust in Bitcoin, the exchange rate would drop, resulting in a smaller real-world gain for the miner. An example of this reasoning is often used to analyze the likelihood of a 50%+1 attack.

50%+1 Attack. We talk about 50%+1 attack when the majority of the hashing power is controlled by a single entity. This entity would be in total control of the blockchain because it could decide to be the only one to add new blocks. To do so it should just ignore blocks published by other miners. This will lead to frequent forks but the branch created by that entity will always eventually win because will

be always backed by the highest hash power. Being the only one to add new blocks, the entity would also be the only one to collect rewards and fees, making mining unprofitable for everyone else, thus driving out miners from the business and hence increasing the entity computational power percentage even more. Controlling the chain, the entity could also decide which transactions to include in the blocks, for example demanding high enough fees or just blacklisting some addresses. It could also delete old transactions, voluntarily forking the chain before the block containing them (called history revision attack), allowing it to perform double spendings. The main argument against this attack is that being the only one to collect new coins, the controlling entity would also be the one to suffer the most from a Bitcoin exchange rate drop. This means that the entity would be encouraged to behave benignly to avoid a drop in public confidence in the protocol that would crash the coins value. Also, the miner would be the most exposed in a Bitcoin crash, due to the huge investment sustained to buy the hardware necessary to reach hash power majority which would then become useless and worthless.

Goldfinger Attack. Another example of a not-reward driven/compliant strategy is when the miner is not driven by the value of Bitcoin earned, but instead they want to achieve some other real-world gain. For example, an attacker would like to perform a 50%+1 attack because they have strong economic interests in a rival cryptocurrency and want to try to scare users off Bitcoin. They also might want to attack Bitcoin just to manipulate the exchange rate, to gain an economical advantage other than just block rewards. An example of those kind of attacks is the so called "Goldfinger attack" [160]. In this attack the aim of the attacker is to destroy Bitcoin, and they are willing to loose money to reach this goal. In this example, the basic assumptions ruling out a 50%+1 attack (anti-profitability and long run losses) would not hold anymore. This attack is also more effective and difficult to model because it relies on lack of users trust in the system which is a difficult parameter to model. A Goldfinger attacker could try to perform the attack even if they control a huge percentage of the computational power, but still less than half of it. With the current six confirmation wait to accept transactions, an attacker with 40% of the computational power will pull a successful history revision attack on a freshly confirmed transaction with a 50% probability, while an attacker with 30% of the total hash power will succeed in the same attack with a 13% probability. It might not seem much, also considering the enormous cost necessary to control such high percentages, but a smart attacker could announce to control the majority of the hash rate and then use some successful attacks to prove it. If performed right, the confusion and disruption created could be enough to drop the exchange rate, thus making mining less profitable and so driving out honest miners. This will increase the attacker percentage, further strengthening their position and the disruption caused until the attack is successful.

1.3.2 Block Discarding Attack

The reward system was thought [195] to be sufficient to encourage miner to behave honestly, under the assumption that each miner gains were only directly proportional to their percentage of the total hash power. Unfortunately this assumption does not hold true in the presence of fraudulent mining techniques. In this section we show how profit driven miners are rationally incentivised to use a fraudulent mining technique.

The underlying idea behind the fraudulent mining techniques under the family of "block discarding attack" [12] is to force the honest miners to waste resources trying to find new blocks on branches that will later be pruned, so *discarding* the blocks mined by hinest miners. To do so, the attacker keeps their found blocks secret and reveals them only to cancel honest miners efforts. The attacker tries to build a secret branch longer than the public branch, leaving the honest miners to work on the public shorter one. This way, when the attacker publishes their secret branch it will be longer, and thus replace the public honest one. This way the total resources dedicated to useful work are lowered, and hence the attacker percentage of blocks found is artificially raised. The simplest example of this technique is called "selfish mining" [109]. The proposed technique is the following:

- If the fraudulent miner knows no secret block, it starts mining on the public chain exactly as honest miners do.
- When the fraudulent miner finds a new block, they do not publish it and instead they keep it secret, then they start mining the next block on top of this secret block. This way they can potentially find a lot of secret blocks and keep building the secret branch.
- When an honest miner finds and publishes a new block, the fraudulent miner can be in four different situations:
  - They do not have any secret blocks. Then they behave as honest miners and simply accept the new block starting mining on top of it.
  - They have only one secret block. Then the rogue miner immediately publishes their secret block to try to win a network propagation race against the honest block. Every miner should mine on top of the first block received if it receives two new different blocks at the same height in the chain. In this case, the miner has some probability (network latency dependent) that the next block found is on top of their block, in which case the honest miners' work is effectively wasted. If a new block is instead found on top of the honest miner's block, the fraudulent miner's block is discarded and no reward is earned by them. Note that each honest miner will mine on top of the first block received, so, depending on network latency, some honest miners will be helping the fraudulent one, mining on top of their block.

- They have two blocks kept secret. Then the fraudulent miner immediately publishes their two blocks, creating a fork before the new honest block or adding them to the previous fork. Their branch now will be the longest, and so the new honest block will be discarded. Then the rogue miner restarts the process from the beginning.
- they have three or more blocks kept secret. The rogue miner immediately publishes their older block, creating a fork or adding it on top of their previous fork. Now the fraudulent public branch and the honest one are equally long, so the honest miners will work on top of one of those two, but any block they found will be useless because the rogue miner already knows at least two more valid blocks on their fraudulent branch and so they will eventually publish those pruning the honest ones.

The first observation about this technique is that it is counterintuitive. A miner is supposed to immediately publish new blocks to collect their reward before some other miner publishes a new block cancelling the first miner's efforts. The fraudulent miner strategy is the opposite. They accept the risk of loosing their certain reward for new blocks, hoping to make the honest miners loose more rewards in useless computations. We also note that the rogue miner has a smaller hash power percentage than the honest miners combined, so its secret branches will eventually be surpassed from the honest miners. This is why it never tries to catch up with a longer branch (which would be probabilistically impossible for them), and as soon as an honest branch takes over, they start to mine on top of it.

Attack Profit. If we follow the calculations presented in [109], calling α the percentage of computational power controlled by the rogue miner and γ the estimated percentage of miners to accept the rogue block in the event of a new block network race, we find that the relative profit of the fraudulent miner is not α as expected, but

$$\frac{SelfishMinerGain}{SelfishMinerGain + HonestMinersGain} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}$$

This is because the total reward is less than one because some computational power (of both honest and rogue miners) was wasted on useless computations:

$$SelfishMinerGain + HonestMinersGain = \frac{\alpha^3 - 2\alpha^2 - \alpha + 1}{2\alpha^3 - 4\alpha^2 + 1} < 1$$

$$if \quad 0 < \alpha < 1/2$$

The relative profit could be greater or less then α, depending on α and γ, in particular

$$SelfishMinerGainPercentage > \alpha \quad iff \quad \frac{1-\gamma}{3-2\gamma} < \alpha < \frac{1}{2}$$

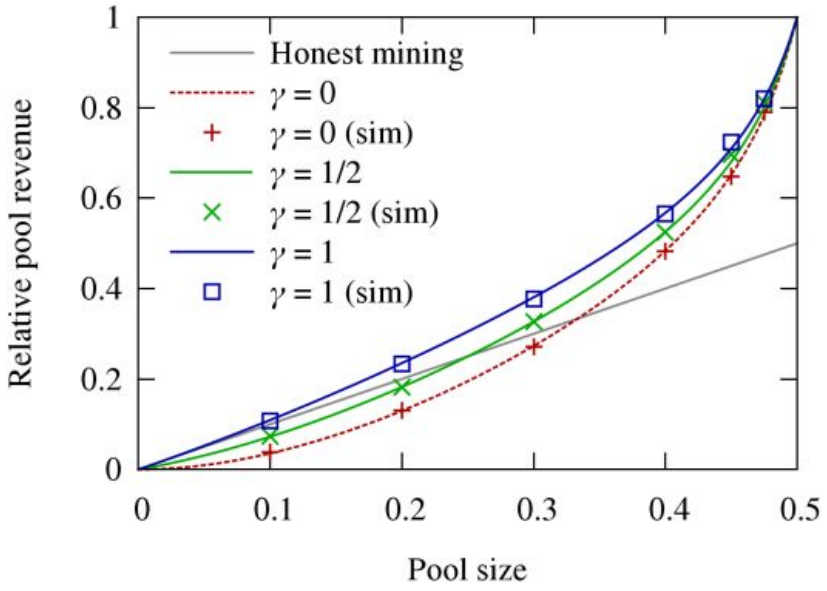


Figure 1.7: Miner revenue using the selfish mine strategy for different propagation factors γ, compared to the compliant mining strategy. Shown both theoretical and simulation results. Source [109].

So the selfish mining strategy is convenient if $\alpha > (1-\gamma)/(3-2\gamma)$. Not only the relative profit is greater than the honestly expected one, but it also grows more than linearly when α increases (see Figure 1.7). Than means that more computational power the rogue miner controls, more unfairly greater its revenues will be. This leads to a scenario in which the rogue miner is incentivized to recruit outside miners to increase its profit, and the outside miners are encouraged to join the rogue mining pool because of the expected higher relative revenues. So as soon as a fraudulent miner passes the profitability threshold (dependent on γ), it will start to attract other miners increasing its revenues until the rogue pool reaches the majority of the hashing power. At this stage the pool controls the chain and it will no longer need to use the selfish mining technique, nor it will accept new members.

Increasing the Profit. We note that the only thing a fraudulent miner could do to increase the effectiveness of the attack is to increase γ. This is theoretically possible if the rogue miner succeeds in being connected directly to all the other miners. In the real network however it is not so simple. Miners usually access the network through one or more gateway nodes and they try to remain as hidden as possible. This is because miners node are the most valuable targets for DDOS attacks. Miners also adopt side channel techniques to communicate between each other to try to reduce the block propagation latency and hence the amount of resources wasted on old branches.

Effectiveness Objections. Some doubts about the claims in [109] are stated

in [65]. The authors state that the rogue strategy profits are only studied in the presence of only one rogue pool against an honest majority, no study has been done about more dishonest miners contemporary adopting fraudulent techniques. Also the idea that the rogue miner should accept outside miners lured in by higher profits to grow the rogue pool until it reaches a majority is not so applicable in practice. The first problem is that the expected profit is theoretically higher than the honest pools, but it can still be irrelevant for real world applications. Also setting up a pool leaves the attacker open to attacks itself. First of all the pool members would notice that the blocks are being kept secret and so they could expose the attacker, also big miners could refuse to partake in a rogue pool scared of the negative popularity effect it could have on the trust in Bitcoin (and so on the exchange rate). Finally, the new members could perform attacks from the inside of the attacker pool (like the block withholding attack explained later), to achieve even higher gains than the attacker.

1.3.3 Block Withholding Attack

This kind of fraudulent mining techniques is based on the mining pools share payment method.

Mining Pools. Bitcoin has seen a huge adoption of miners coalitions called mining pools (see Figure 1.8). A mining pool is a centralized or distributed infrastructure comprising a set of miners working on the same PoW. In centralized pools the pool manager is the one in charge of distributing the revenues among the miners. Miners are not only payed if they find a block, but also for the effort spent on failed attempts. To keep track of the effective effort spent each miner sends to the manager not only valid PoWs with the current difficulty but also solutions with lower difficulties. This proves the manager that the miner is actually working at the PoW but was not lucky enough to find a valid solution. The rewards collected by the pool from valid blocks mined are divided among all the pool participants proportionally to the computational power used, as proved by the lower difficulty solutions submitted. It is worth noting that joining a pool does not increase a miner expected profit, which is still proportional to the miners hash power (if it is an honest miner). The expected profit is instead slightly lowered because of the fees taken from the pool operator. The first advantage of joining a pool is a practical one, because the miner does not have to care about transaction listening and block creation, it has just to try to solve PoWs provided by the manager. The major economical advantage of joining a pool is that the expected variance is significantly decreased. The miner is expecting to be paid a little less regularly rather than collecting one big reward once in a while.

Stealing Pool Shares. The block withholding attack [65] is based on the idea of collecting the shares participating in the pool without actually increasing the pool computational power. As we have already seen with block discarding attacks,

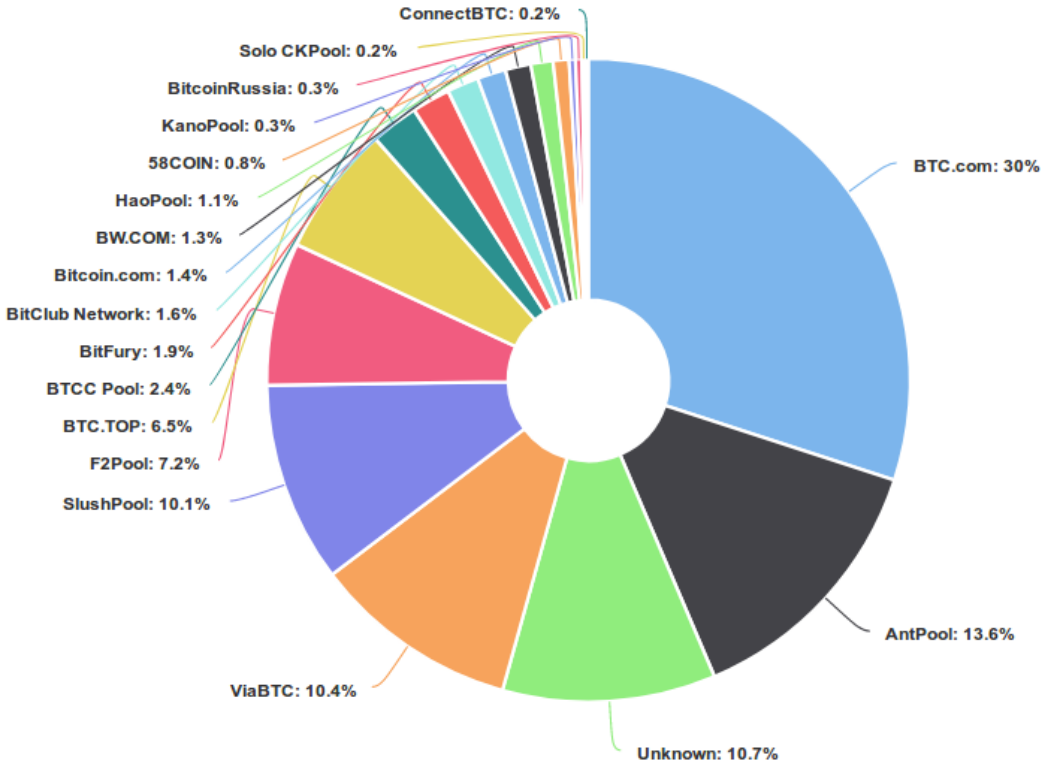


Figure 1.8: Estimation of hashrate distribution amongst the largest mining pools over the last four days from the 24th of April 2018. Source [38].

the aim of the attack is to fraudulently increase the rogue miner profit lowering the honest miners profits. This allows the rogue miner to gain a percentage of profits larger than its hash power percentage. To perform the attack, the miner splits its computational power with two different purposes. The optimal split is to divide exactly in half the hash power. Half of the power is used from the miner to honestly mine for new blocks. The other half is divided among the other honest mining pools. The infiltrated attacker controlled hash power correctly tries to solve PoWs assigned from the honest managers and sends them the solutions with lower difficulty to collect the shares but never reveals (withhold) valid blocks found. Note that the found blocks are useless for the attacker because they are built from honest managers who would be the one to collect the rewards.

1.4 Double Spending

Transaction Confirmations. We have already explained that the Bitcoin consensus protocol is proven to reach consensus at any time only on a common prefix of the blockchain, the most recent blocks could be part of a fork during which no official

accepted blocks are defined. During a fork two (or more) transaction histories exist at the same time, so the same funds could be spent twice, once on each history. Only one history will eventually prevail, deleting the conflicting transactions in the pruned branches. But if those transactions were accepted and the real world benefits had already been gained, then a double spending attempt was successful. Bitcoin solution is to wait a fixed number of "confirmations" before accepting a transaction in the blockchain as valid. The number of confirmation indicates the minimum number of block the user has to wait to be mined on top of the block containing the transaction. This number is arbitrary set to six by default, but can be increased or decreased depending on the value exchanged by the transaction [234]. Increasing the number of confirmations to wait increases the probability that the transaction will be included in the final valid transactions history, but it also requires the users to wait for more time. In fact, on average, a new block is validated every ten minutes, so a six confirmation rule already requires the user to wait one hour, on average, for every transaction.

Fast Payments Attacks. This waiting time is too much for a lot of real world payments scenarios, where users can wait only seconds for a payment. This has brought to the adoption of "fast payments". A fast payment is validated using rules which do not consider the blockchain. This obviously makes fast payments transactions vulnerable to double spending, because the blockchain is the solution adopted precisely to prevent double spending. The majority of fast payment double spending prevention techniques are based on a listening period on the network to detect double spending attempts [13]. If a transaction reaches first the majority of the miners it will be mined in a block and so no double spending is possible. Because of the Bitcoin fast message propagation [73] there is only a limited time window during which the transaction is still traveling and has not reached all the miners. This time window is what an attacker should exploit to pull a double spending attempt. The goal of the attacker is to let the majority of the miners to know its fraudulent transaction before the honest one without the victim noticing. Due to the gossip message flooding, if the attacker manages to make the fraudulent transaction to reach the majority of the nodes it will also reach with high probability the honest node, which will become aware of the attack. The attacker can use smarter techniques, like for example using the victim neighbors as a screen to isolate the victim from the network. It can also combine the double spending attack with more powerful techniques like a sybil attack, to feed to the victim a different fake reality. The victim can try to protect itself deploying listener nodes in the network or using a listener service. In the end the attack reduces itself to a stealth race between the honest and fraudulent transaction. This kind of attack is studied in [151].

Finney Attack. If the attacker is also a miner it can perform a more powerful attack called "Finney attack" [114]. The attack is more costly because requires the attacker to invest a lot of computational power in the mining attempt, but it is also

much more powerful because it renders any listening countermeasure useless. The base idea of the attack is that the attacker tries to mine a valid block containing the fraudulent transaction. As soon as it succeeds, instead of immediately publish the block as usual it broadcasts the honest transaction instead. The victim cannot know of the secret block and no conflicting transaction is ever published on the network, so it will accept the transaction. As soon as the transaction is finalized by the victim the attacker publishes the secret block, which is valid and so will be accepted by everyone, rendering the honest transaction invalid and so discarded. The double spending was successful.

The main advantage of the attacker is that it can first try to mine the block and only when it manages to mine it, it can launch a double spending attempt, this means that the miner has no hurry to find the block and so also miners with little percentages of the hash power can perform the attack. The main problem for the attacker however is that it has to wait for the victim to finalize the transaction. During the time which passes between the finding of the block and the victim acceptance some other miner could find a valid block which will render the secret block invalid. So the attacker is not only risking to see its double spending attempt failing, but it is also risking to loose any reward granted from the block. Usually the block reward is much higher than any sum in a transaction which is accepted as a fast payment, so the attack would not be economically rational.

1.5 Deanonymization

Since the very start [195] users anonymity was not Bitcoin main goal. This has brought to a major flaw in the project, which protects user anonymity only with the use of pseudonymity, with the hope of breaking ownership linking between addresses and real world identities. Any cryptocurrency who wants to scale to world acceptance needs strong privacy assurances, which nowadays Bitcoin seems to fail to give.

While studying Bitcoin privacy we should always remember three important properties of the protocol:

- The entire transaction history is publicly available and persistent in the blockchain.
- Anonymity is enforced only through pseudonymity, which means that each user partake in the protocol only through an arbitrarily large number of addresses that carry no real-world information about the user (nor any information between themselves).
- Transactions are broadcasted to the network (mostly) by the creators (gossip), every user has to create is own transactions, so it is also the first one tasked with informing the other peers about that new transaction.

Formal Anonymity Definitions. To analyze the effective anonymity provided by the protocol, we should first formally define which properties are requested to enforce anonymity. Few works are available on this topic, the most notable being [5], where two different properties to model Bitcoin privacy are presented:

- Activity (address) unlinkability. It refers to the fact that an adversary should not be able to link two different addresses pertaining to a user of its choice [5].
- Profile indistinguishability refers to the (in-)ability of an adversary to reconstruct the profiles of all the users that participate in the blockchain. Profiles, here, consist of the set of addresses (address-based profiles) or set of transactions (transaction-based profiles) of Bitcoin users [5] .

Other efforts on formally defining anonymity properties are mainly present in works also presenting solutions to enforce those same properties, as for example [185, 238, 70].

We will now analyze the state of the art of the main known attacks on Bitcoin privacy.

1.5.1 Network Listening

The most simple form of network listening is to just deploy one or more listeners in the network recording the message flow. This approach is not a threat to the network and allows for information propagation studies and basic topology findings [35, 91]. The most cited example is [73] where transactions and blocks propagation times are studied and real network data observation are used to prove how block propagation delay influences blockchain forks.

Attack Goal. The goal of network listening based attacks is to link network information with Bitcoin addresses [158, 113]. Obtaining so we have succeeded in connecting some real world information to some addresses. We note that by saying "network information" we do not just mean an IP. IP addresses are not enough information to correctly represent a single user when multiple users are behind the same NAT or employ an anonymization layer like TOR. We need a way to assign to each user a "network signature" to differentiate users. The most effective proposal was presented in [22] where the authors propose to use the "entry nodes set" as signature of each peer, where the entry nodes set is the set of nodes (eight by default) a peer initially establishes its outgoing connections to, each time it connects to the network. This set is not network dependent and sufficiently random to differentiate peers behind the same proxy.

The most studied network deanonymization attack is based on the assumption that the first node in the network to broadcast a new transaction is the creator of that transaction, which is also the owner of the input addresses of that transaction, because in order to create the transaction it had to sign it with the corresponding private keys.

This assumption is true for the vast majority of transactions, but is not necessary true for some special transactions. An example is given by CoinJoin transactions [177], which are single transactions involving multiple users, therefore the inputs are not owned by a single signer.

Attack Scope. In general, automated attacks could always be deceived by active users countermeasures, so any attack should always try to balance false positives with recall. There are in general two kind of attacks, the general attack and the targeted attack. In the general attack, the attacker mostly uses general rules to avoid as much false positives as possible, accepting a low recall rate, acceptable in the presence of a huge amount of data. The aim of the attacker is to break the profile indistinguishability property. In the targeted attack the attacker only considers a relatively small set of addresses and it tries to find at least one linking between one of those addresses and some real-world information. The attacker is trying to break the weaker privacy property of address unlinkability, trying to build an address chain of ownership. An example of attacker for the general point of view could be a researcher trying to gather general information on the network as a whole [35, 91, 180, 22, 187], while an example of targeted attacker could be a law enforcement agency who want to find a link between funds involved in illicit activities and a real world entity [180, 230, 113].

State of the Art. An example of network listening to link IP with transaction creators (and so addresses) is presented in [158]. The deployed listener simply tried to connect to every network address advertised on the network, keeping alive the connection indefinitely [35, 91]. The listening period lasted five months from July the 24th 2012 to January the 2nd 2013, with a median of 2678 active connections per hour. The analysis was based on the standard assumption that the first node to rely a transaction in the network is its creator, but since no refined technique were employed the simple listener could not know if the first node it received the transaction from was also the first to release it on the network or was just propagating it. So the analysis was mainly relying on anomalous relaying behavior, which for example includes transaction being relayed by a single peer (and not by others because, for example, are non standard or double spending attempts) or transactions relayed more than once from the same peer (in the official client a peer sends a transaction again to a neighbor only if the transaction is not confirmed yet and the peer is the sender or a recipient in that transaction). In the end, the study allowed to associate a probable IP to 1162 addresses over four millions transactions collected.

A more refined attack was explained in [22]. The technique is based on noticing that a peer is uniquely identified by its entry node set (as explained before). First of all the attacker tries to find the entry node set for a peer when the peer connects to the network for the first time. When the peer connects it will forward its own address to its eight entry nodes. So the attacker remembers the first nodes from whom it has heard of any new address. Those eight nodes will not be the correct entry node set because of network latency and random message forwarding, but

with high probability there will be a lot of nodes in common between this attacker estimated set and the actual entry nodes set. Then the attacker considers the first n (greater than eight) nodes each transaction is first received from and tries to find an estimated entry nodes set correlated to those n nodes (for example with at least three nodes in common), in order to associate the transaction with the peer identified by that set. An attacker can increase the success probability by having a very high number of connections with each peer (in the paper is suggested to have at least fifty connections with each full node), because that increases the probability to be chosen as first node to relay a transaction to. With fifty connections to each peer and three tuple matching as correlation measure, the experimental results in the Bitcoin testnet show a success rate of 60% and in the real network the success probability is estimated as 11%. We nevertheless remark that the attack presented is really invasive and disruptive on the real network. Each node keeps by default a maximum of 125 active connections, so an attacker establishing 50 connections to each node would consume almost half of the connection slots available, resulting in a quite heavy DoS on the network. A main advantage of this technique is that it allows for the first time to differentiate nodes behind NAT or firewalls (or in general nodes not accepting incoming connections) introducing the entry set recognition.

Avoiding TOR countermeasure. In [22] a technique to exclude TOR (or others similar anonymity proxies) nodes from the Bitcoin network is also presented. The basic idea is to ban TOR exit nodes relaying malformed Bitcoin data through them to have them blacklisted by honest nodes as DoS countermeasure. This allows an attacker to prevent honest users to connect to the Bitcoin network via TOR, but the attack is highly detectable. In a subsequent paper [23] the same authors show how using Bitcoin over Tor opens new possibilities for man-in-the-middle and sybil attacks. In [23] the authors also present a smart fingerprinting technique to recognize users, based on attacker chosen network addresses injected in the peers network addresses lists.

Topology Discovery

A network listener attack always relies on one or more listeners deployed on the live network, with as many connections to honest peers as possible. The Bitcoin network is a pure P2P network with no topology overlay. This means that the only information nodes have about the network topology is the set of their neighbors. Nodes also do not exchange directly any information about their neighbors set, so topology cannot be easily learned. Of course knowing the topology will highly benefit a network listening attack as well as other kinds of attacks, so some research has been directed in topology discovery attempts. The two most promising ideas are presented in [187] and [22].

State of the Art. In [187] the authors present a topology discovery technique based on network addresses timestamps. The attacker needs a direct connection to

each node it wants to find the set of neighbors because it needs to directly send messages to it (known network addresses list requests). The attacker then asks for as many network addresses known by the target as possible. The underlying idea is that the protocol rules dictate a node to age the timestamp of all the nodes it knows about without being connected to them and keep unchanged (or up to date) the timestamps associated to directly connected nodes. So, by analyzing discrete two hour differences in timestamps, the attacker can obtain the set of neighbors of each node. We now should point out that the technique worked at the time of the paper but is based on protocol rules of the time who may change (and indeed have changed today). This kind of exploit is based on current implementation rules and not on immutable protocol principles. The technique presented is a good example as today is not usable anymore, with version 0.10.1 of the default client the timestamp policy was changed rendering this attack unusable. In the paper [187] is also present a more general attack to discover mining pool influential nodes, called by the authors *decloaking*. A mining pool usually interfaces to the network through one or more gateway nodes, nodes tasked with collecting transactions and other miners blocks and relay the newly found pool blocks. The gateway nodes behave as a normal full node and it is important for the pool that they remain anonymous. This is because if a gateway node is discovered it can be targeted with DDoS or Sybil attacks, damaging the pool revenues. The technique presented consists in sending a different marker transaction to each node we are connected to, each transaction must be conflicting with the others so that only one can be accepted in the chain. Then we check the blockchain to see which one was validated. Repeating this experiment we can find the nodes that most often get their marker transactions validated. Those are the nodes closer to mining pool gateways (or maybe are gateways themselves). This simple method was proven effective in the paper where is shown how the top 100 influential nodes found accounted for three quarters of the validated transactions.

In [22] is present another topology discovery techniques to learn full nodes (or any other type of node accepting incoming connections) degree and neighbors. The underlying idea is to send marker addresses (with a timestamp which allows it to travel only for few hops, hopefully one, under the current relaying rules) to a peer which will then forward some of them to its neighbors. If then the attacker queries the other nodes (neighbor candidates) for the marker addresses (the attacker learns most of the node known addresses and then check how many are markers) it can estimate the probability that they are neighbors of the target node (based on the number of known marker addresses).

Malleability

A side effect of a listening attack is that it also provides the attacker with all the live information circulating on the network which are later forgotten and never recorded in the blockchain. These live data can be very useful to extract more information from the network. An example of this approach is presented in [74], where the

authors use the live data to estimate the actual amount of malleability attacks in the network.

Malleability Definition. When a transaction is created it is signed and the signature is attached to the transaction but the signature itself cannot be signed. This means that anyone can modify the signature representation (just its representation, not its content) without invalidating it, in this case we say that the transaction was "malleated" [74, 6]. Usually transactions are represented by their hash which is used as the transaction id. If a user malleates a transaction it will change the transaction hash and so its id.

Malleability Attacks. Transaction malleability can be used to perform particular kinds of double spending attempts. We have a double spending attempt because there are two different transactions released on the network at the same time, so there is a conflict, but an important point to remember is that the meaning of the two transactions is the same. Only the signature representation is modified, the addresses and amounts transferred are identical, so whichever of the two gets validated there will be no difference for the addresses balance involved[5].

This double spending attempt becomes dangerous if used against a bad designed software who relies only on transactions ids. Some third party software checks if a transaction issued by them is accepted on the blockchain only looking if the corresponding transaction hash is included in the chain and not the transaction itself. Exploiting this fact, an attack becomes possible. A fraudulent client can ask the id based service for a payment and then malleate the newly issued transaction as soon as it is released on the network by the service. The two transactions will conflict with each other and so only one will be included in the blockchain, starting a validation-race. If the original transaction is included the attack fails. If, instead, the malleated transaction is included in the chain then the attacker can complain with the service that it never received a payment. The service will check the chain for the transaction id it knows without finding it and so will believe the client. Of course if the service tries to publish the transaction again, even modified by increasing the fees, it will be always rejected because it will be seen from the other peers as a double spending attempt. This is because the funds were already spent in the validated malleated transaction. The attack is successful because the attacker has been paid while tricking the service in believing it has not. Eventually the service will emit a new transaction spending new funds to pay the fraudulent user who will so be paid twice (or even more times if it succeeds in malleating also this new transaction). The entire attack is based on the service kept balance history being different from the globally blockchain stored history, so the service does not know of payments already finalized by the network.

Malleability Examples. A big scandal arose when the biggest Bitcoin exchange,

[5]The addresses and amounts are signed and so cannot be modified without invalidating the signature. The only parts of the transaction not signed, and so modifiable, are the signatures themselves.

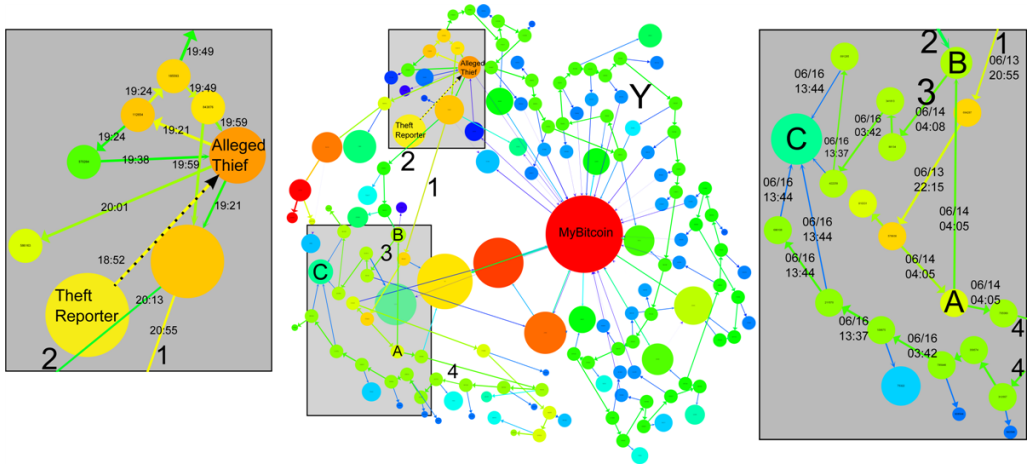


Figure 1.9: Theft tracking example about the famous *allinvain* theft of 25,001 BTC happened the 13th of June 2011. Source [113].

Mt.Gox [66], declared bankruptcy bringing a malleability attack as motivation for the users funds loss. In
[74] a single highly connected listener was deployed in the Bitcoin network to collect transactions live data, looking for malleability attacks. The results show how the Mt.Gox claims look difficult to believe as only 1.811,58 BTC were involved in malleability attacks from the analysis start until the service lock down (between 1/01/2013 and 7/02/2014). This example clearly explains the huge opportunities provided by live network data analysis.

1.5.2 Blockchain Deanonymization Attack

The most studied kind of deanonymization attacks is the blockchain analysis attack. The aim of this attack is to break the profile indistinguishability property by aggregating Bitcoin addresses in clusters, each representing a single user, using the historical information stored in the blockchain. The attack is usually strengthened by also using external information to link each address cluster with some real world information to deanonymize users.

Theft Tracking. We note that also benign attacks and positive results are possible through deanonymization attacks. An example is theft tracking (see Figure 1.9). In Bitcoin transactions are irreversible and so theft can be publicly reported but can not be reversed. The stolen funds can anyway be blocked if honest users refuse to accept them as payments. Of course strong evidences need to be presented in order to avoid DoS attacks with fake thefts. The deanonymization techniques allow also to keep track of the stolen funds preventing the thief from laundering them through obfuscation techniques [180].

Attack Scheme. The attack can be divided in three steps (shown in Figure 1.10):

- In the first step the attacker passively parses the blockchain to obtain a transaction graph. The transaction graph is a graph where each node represents an address used in the blockchain and each oriented edge connecting two nodes represents an atomic transactions involving those two addresses and can be weighted (for example with the transaction timestamp and value). With atomic transaction we mean that we split each multi-input multi-output transaction in more atomic transactions, each connecting one of the inputs with one of the outputs, so that each output has exactly one ingoing atomic transaction.

- In the second step the attacker performs a heuristic rules based clustering applied to the previous graph to obtain a so called "users graph". In this new graph the nodes are now sets of addresses all controlled by same user (see Figure 1.11 for an example).

- The final step is the users graph deanonymization using external information to obtain an identities graph. The techniques used to perform the real world identity-user linking are various but always based on external information gathering [113]. This can be done crawling the web to search for Bitcoin addresses used in posts or donations signatures. Of course voluntary disclosed information are less reliable and so should be accordingly weighted. A more reliable method is to directly interact with services and well known economic parties to learn their addresses through economic exchanges, as performed in [180]. This step can also be combined with a network listening attack to use the listening results to perform identity linking. Is worth noting that this step can also refine the results obtained at the previous step, further collapsing together user clusters.

The intermediate step is the most important and difficult one of the three. Addresses clustering is the step that breaks address unlinkability and hence should be carefully planned. The clustering is based on heuristic rules i.e. rules based on Bitcoin users behavior and common practice observation as well as technical constraints. These rules are time dependent and not general and can lead to a high rate of false positives or negatives if not correctly tuned. Of course it is preferable to decrease false positives at the expense of an increase in false negatives to avoid flawed final results, but false positives are inevitable. Basing ourselves on general rules to model users behavior we inevitably misinterpret some uncommon use cases. This is why in the literature only two heuristic rules are always used and generally accepted.

Multiple Inputs Heuristic

The first heuristic proposed [230, 180, 113, 5, 201] and the one considered the safest and most used is called "multiple inputs". The heuristic rule is the following:

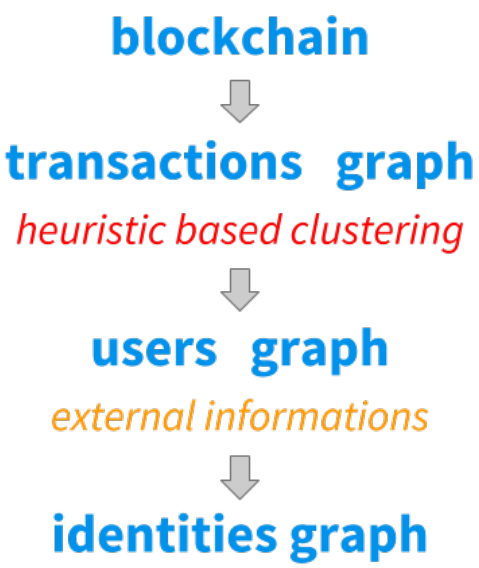


Figure 1.10: Scheme of the presented Bitcoin deanonymization attack.

> *"in a multi-inputs transaction all the input addresses belong to the same user"*

Every transaction must be signed for every input, it means that to sign the transaction the private key of each input should be known. This implies that the signer knows all the private keys and hence it is the owner of all the input addresses. This first simple heuristic rule has been used with success in the past but it is no longer valid for all transactions. A new countermeasure has been designed to trick the "multi inputs" heuristic.

CoinJoin and CoinShuffle. The first effective countermeasure proposed was CoinJoin [177] which underlying idea is for a set of users to meet and collectively create a single transaction using their addresses. First several users agree to take part in a new transaction creation through a meet-up server or decentralized consensus. Then they agree on fixed sum input transactions to be joined in a single big transaction. This means that each user uses one input address with a fixed sum and a single new output address to collect its payment to itself. The transaction must be signed by all the users to be valid, so each user can abort the transaction at any time. This prevents thefts but leaves the protocol open to DoS attempts. An attacker can join as many transactions as possible and then always abort refusing to sign them. Another problem is that each information about other participants is shared among all the users creating a transaction together. This means that "internal linkability" is possible and other information, such as IP addresses, might be leaked.

An improved protocol providing internal unlinkability called CoinShuffle [235] has been proposed, but it also suffers from DoS attacks, because it allows to detect

a party that aborts but cannot prevent fraudulent parties from aborting. We also remark that the privacy offered by the common transaction is limited to the number of users taking part in that transaction, so we talk about k-anonymity. The anonymity set size can be increased chaining together more joined transactions, to allow the anonymity set to include all users partaking in all the joined transactions in the chain.

Change Address Heuristic

The second most used heuristic rule [180, 113, 5] is called "change address heuristic". The heuristic rule is the following:

"the change address belongs to the same user of the input addresses"

The rule looks simple but the main problem is to correctly identify which address is actually the change address. This means that the rule yields a lot of false positives and so must be refined to decrease them (at the expense of the recall rate). The refinements are heuristic sub-rules themselves based on users behavior and current wallets software implementation. Some of these sub-rules adopted in [180] are presented here. The address c is a change address in transaction t if and only if:

- t is not a coinbase transaction
- $\mid outputs(t) \mid > 1$
- $inputs(t) \cap outputs(t) = \emptyset$ (no self change)
- there is no address already labeled as change address or belonging to the inputs owner in $outputs(t)$
- it is the first time c appears in the blockchain and there is no other address in $outputs(t)$ satisfying this condition

Countermeasures

We have already seen CoinJoin [177] and CoinShuffle [235] proposals to trick the heuristic driven clustering attack, but more general techniques are used to try to protect against blockchain analysis. All of the techniques proposed try to break the ownership tracking among transactions.

Obfuscation Techniques. The most simple techniques used are called obfuscation techniques [230] and consist in scrambling funds among user controlled addresses to confuse the attacker. The used techniques rely on well known randomized patterns, the most notable being splits and aggregations, peeling chains, and binary trees. All those techniques succeed only in creating confusion, without effectively breaking the funds ownership and so can be easily detected by a motivated attacker [230, 180, 113].

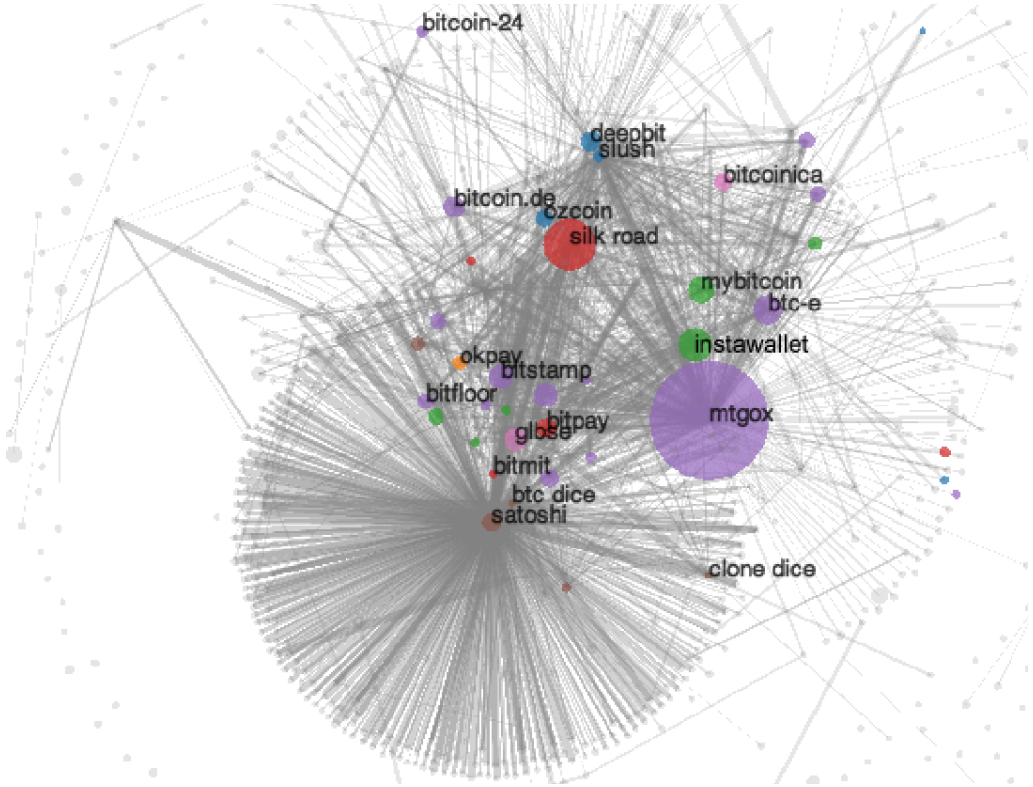


Figure 1.11: The Bitcoin users graph (with clusters partially labeled), obtained using both the two heuristic rules presented. The area of the cluster represents the value received from other clusters but not itself, and for an edge to appear between two nodes there must have been at least 200 transactions between them. The nodes are colored by category: blue nodes are mining pools; orange are fixed-rate exchanges; green are wallets; red are vendors; purple are (bank) exchanges; brown are gambling; pink are investment schemes; and grey are uncategorized. Source [180].

Mixers. A better countermeasure is to use a mixer, which is a third party entity collecting standard-sized payments from customers directed to a set of input addresses and pays back the same clients at random times in the future using funds stored in addresses not linked to any of the input addresses by any transaction. This effectively breaks ownership tracking but leaves the user exposed to thefts from the third party mixer, which needs to be trusted. The mixer also knows the internal linking of users, allowing internal linkability, and users need to pay a fee to use the service. Laundering the funds through a chain of different mixers will increase the anonymity but also the cost and the risk of thefts. A review of mixing services is presented in [192] where is shown as the BitLaundry mixing service does not actually break ownership tracking.

1.5.3 Simplified Payment Verification

To use the explained protocol a user needs to know the entire blockchain (or its compressed form). For occasional users who use Bitcoin only to pay small sums through smart phones, downloading the entire blockchain is impossible. To cope with light weight users the Simplified Payment Verification (SPV) was defined [195].

We firstly note that the chain is used to check the validity of transactions received from other users and not to create new transactions, so a SPV user will create and broadcast transactions as everyone else.

When an SPV user needs to check for a transaction validity (containing for example a payment to it) it sends to full nodes a special SPV addresses request containing the addresses the SPV user is interested in. When the full nodes learn new transactions containing the requested addresses they forward the transaction to the user with the merkle tree of the block containing the transaction and the header of all the blocks until a certain depth in the blockchain. The merkle tree allows the user to check that the transaction is really included in that block, while the block headers allow the user to check that the block is part of the longest chain. It is worth noting that the block headers are constant in size and much smaller than the entire block (81 bytes against a variable size up to 1 MB). We also note that the user cannot check the transaction validity because it does not know the other transactions. For example it cannot check that the input used in the transaction are really present in the chain. But the user knows that the block containing that transaction is part of the longest chain and this means that the majority of miners have worked on top of that block and so they deemed the transaction valid.

Bloom Filters. Of course the fact that the SPV user lets the full nodes know which addresses it is interested in is a clear privacy problem. To avoid direct disclosure the SPV nodes send the full nodes only a Bloom filter with a desired false positive rate, this allows the full nodes to check if an address matches with the filter without directly knowing the user addresses. Unfortunately a lot of information is leaked by those Bloom filters, resulting in a big privacy issue for SPV users. In [123] the author shows that a lot of addresses can be recovered from one filter of a user with a little number of addresses. They also prove that an attacker can easily discover which different filters are created by the same user and that an attacker can learn a lot of addresses of a user if it knows at least two different filters of the same user. The SPV deanonymization can be combined with the previous techniques to strengthen the obtained results.

1.6 Ethereum

In this section we provide a basic explanation of the Ethereum protocol through a very high level presentation, omitting most of the implementation details not relevant to this thesis. For a more detailed explanation see [276].

Ethereum starts from a simple assumption, coupling a Bitcoin style cryptocurrency (see [84]) with Turing complete applications. The protocol can still be seen as a cryptocurrency due to the *Ether* currency on which it is based. The way in which value expressed in Ether is managed is in principle not too different from the way Bitcoin manages value explained in Section 1.2. The system is based on pseudonymous entities (masked by addresses) that exchange value through special data structures called transactions that are broadcast to the underlying communication network. Those transactions are eventually validated and will then apply their effect by updating the global state (recording the passage of value to the new owners). Do note that Ethereum uses the concept of *accounts* rather than Bitcoin addresses. Accounts store the current balance and transactions directly update such value in the current state,differently from Bitcoin where transaction outputs actually remembered the value stored in each address.

Since the reference scenario is a trustless decentralized system, a distributed consensus algorithm is needed to reach an agreement on which new transaction to deem valid. Same as Bitcoin the distributed consensus algorithm used is currently based on PoW (i.e., Proof-of-Work) and it is called *Ethash* [100], even if the algorithm chosen is *memory hard* to try to reduce ASIC efficiency and there are proposals to move to different consensus algorithm schemes, the more advanced one proposing an hybrid scheme obtained by an integration of PoW and Proof-of-Stake (PoS) [101, 51]. Exactly as Bitcoin, the pending transactions are grouped together by miners into blocks, one of which will eventually be securely added to the globally accepted chain of blocks, that determines the updated global state.

The novel contribution of the Ethereum protocol is to use the blockchain not only to execute value transfers but also code. In fact if we think of the Bitcoin blockchain we can see it as way of representing a shared state, updated by transactions. The states only records the amount of value coupled with each address. If we drop this limitation, and consequently allow the global state to store any variable to value coupling, we obtained a shared memory for an hypothetical virtual machine. We can then use transactions to allow for arbitrary complex code to update such state. The obtained system is, of course, more powerful than the Bitcoin state machine, since peers agree on the state obtained by arbitrary computations, but the underlying way of functioning, i.e. sequentially executing transactions as dictated by their ordering in blocks, is the same.

In practice, alongside traditional value exchange transactions, users can also create transactions carrying executable Turing complete code. Those pieces of code deployed on the blockchain are called *smart contracts*. A transaction carrying the payload of the contract is first broadcast to the network. Its result is the deployment

of the payload as code linked by its public address. Any new transaction can then refer to this address to trigger the execution of the functions inside the contract, carrying the function parameters inside the transaction payload and showing the eventual return value to the outside. Basically a smart contract contains some storage space for its data and some callable functions. Smart contracts can themselves act like regular users creating new contracts and sending payments or function calls between themselves. This means that in Ethereum there are two types of accounts:

- *Contract Accounts*, holding smart contract data;
- *Externally Owned Accounts*, representing traditional users controlled accounts.

Validating (i.e., adding it to a block) a transaction containing code (either calling a contract function or deploying a new contract) means to execute such code, updating the global state accordingly. As explained in the previous paragraph, this means that the global state can be seen as a virtual machine running all the code on the blockchain (called *Ethereum Virtual Machine* or *EVM*). At the same time all the miners adding a transaction with code to a block actually execute that code, so the code execution is replicated between all the miners. Since all miners have to agree on the same result of the execution the EVM needs to be fully deterministic, otherwise no consensus could be possible. Do note that transactions can only be created by users and so only originates from Externally Owned Accounts. Despite that, once a Contract Account is awakened by a transaction it can subsequently communicate to other smart contracts through *messages* who carry the same logic as transactions, but are an effect of transactions, rather than being transaction themselves.

Often smart contracts are coupled with the term *dApp* [71], short for *decentralised app*. A dApp is simply an app with the backend (or server side) is implemented as one or more smart contracts on the blockchain. Although such apps can be accessed directly through blockchain transactions to the relevant smart contracts, to increase users fungibility and ease users interactions, they often offer an user friendly frontend, usually web based, also named *dApp client*.

It is important to remark that every transaction has to pay a fee proportional to its complexity to repay the miners of their effort of maintaining the EVM. To achieve this, to every single operation of the EVM is assigned (by the protocol, see [276]) a price proportional to its burden to the users (i.e., the number of computational steps needed for its execution and its storage weight), this is called *gas* and the total gas of a transaction is the summation of all the gas of every single instruction it contains. This is the gas that is consumed by the transaction upon validation. The entity (either a user or a contract) creating the transaction needs to decide two parameters, the *gas limit* and *gas price*. The gas limit is the maximum amount of gas the transaction is allowed to consume, if it is exceeded all gas is spent but the execution effects on the state are reverted. This is useful to avoid too long or even infinite computations that would stall the EVM. Furthermore each block has associated a *block gas limit* to guarantee a limit to the amount of computation

executed by all the transactions in that single block. The gas price is instead set by the user as the amount of Ether the user is willing to pay for each unit of gas. Miners are free to choose what transaction to mine and so can refuse the ones with a gas price too low.

The smart contracts are written in a low level bytecode language interpreted by the EVM, however, high level languages whose programs can be compiled in EVM bytecode (producing a *.bin* file containing the binary of the compiled contract and an *.abi* file containing the contract interface specification) have also been developed to ease human smart contract coding. The most widespread of such languages is a JavaScript style language called Solidity [247], presented in Section 1.6.1.

1.6.1 Solidity

Solidity [247] is a programming language to write smart contracts, it is used mainly by the Ethereum project, but not only (e.g. *Monax* [189], and the *Hyperledger Burrow* project [138] of which Monax is a partner). In fact, it is one of the five languages (the others being Serpent [240], LLL [170], Viper (experimental) [272] and Mutan (deprecated) [194]) developed to be compiled in EVM bytecode to be run by the Ethereum Virtual Machine. It was initially developed and proposed by Gavin Wood [248] and other Ethereum core developers in 2014. The language development was then carried on by the Ethereum project's Solidity team, led by Christian Reitwiessner [245].

For an example of a Solidity written contract see Listing 8.2. Solidity syntax is mainly inspired by JavaScript (but influenced also by C++ and Python), its *contract oriented* programming style is very similar to classical object oriented programming. In fact, a solidity program defines one or more contracts, where each contract can be seen as a class with certain methods (i.e. contract functions) and local state. Do note that by *local* state we mean that such state is only modifiable from within the contract through its defined functions, but such state is still globally readable, since it is publicly stored on the blockchain. In fact a contract can not keep any private information, otherwise any user would not be able to execute it and deterministically obtain the same result.

Solidity is statically typed and supports multiple contract inheritance but allows for a single constructor to be defined for each contract. Each contract can contain definitions of *state variables*, *composed types*, *functions*, *function modifiers* and *events*.

State variables are used to store the local state, they have a statically known type and visibility modifiers. The supported types comprehend the usual basic types, such as booleans and integers as well as novel data types, such as Ethereum hexadecimal addresses (see [244] for a complete list). Complex types such as lists and maps are supported as well, and composed data types can be defined through *enums* and *structs* of finite size.

As already explained, all data stored in the contract variables are saved on

the blockchain, and so visible to any user, this means that visibility modifiers for variables are only useful to define visibility between contracts. State variables can be defined *public* (visible from any contract), *internal* (visible only to this contract and the ones inheriting from it) or *private* (visible only to the contract they are defined in). Notably, the solidity compiler creates a public *getter* function (i.e. a function simply reading and returning the variable value) for all variables declared public inside a contract, to make the corresponding values easily readable from the outside.

Events are a special way of storing information in the blockchain without using the contract local storage, they can be seen as a way to define a special message structure. An event can be fired by a function inside a contract, causing the relative message to be stored in a logging storing space in the blockchain associated to the contract. Events are cheaper than normal data storage operations, but, since data is not stored inside the contract local state, they are not accessible from any contract. The advantage of using events is that they can be indexed and queried by clients to easily retrieve information or being notified for something happening. They should be seen as a logging or alert one way communication channel for contracts to notify clients.

Functions codify the main functionalities of a contract, and they can have different visibility. They share the same visibility modifiers of variables (*public*, *internal* and *private*) with the addition of the *external* modifier, that allow a function to be visible only outside this contract, so it can only be called by the same contract through an external message to itself. Since only external and public functions are visible, and so callable, from the outside of the contract, they are the only ones that can be called through transactions (or other contracts messages). In fact, the compilation of a contract produces an *application binary interface* (ABI) defining the contract function signatures callable from the outside (either external clients or other contracts). Functions fall in two main categories, the ones that modify the state and the ones that do not. In fact functions that do not require to modify the state (e.g. the variable getter functions explained before) can be directly executed by the clients without the need for any kind of consensus and so, without paying any fees. Moreover, each contract may define a single constructor function, containing the code to be executed upon deployment.

Function definitions can be enriched by adding to their signature one or more function modifiers visible in the contract. A function modifier defines a snippet of code to be executed before the function main body, and are often used to check for certain properties before function execution. Multiple modifiers can be added to a single function to be executed in the declared order.

2

Users Graph Analysis

Abstract

In this chapter we present our set of analysis performed on the Bitcoin users graph. We first explain the efficient clustering algorithm used to obtain the graph from the raw blockchain data. We then perform a wide set of classical graph analysis. In particular we perform a connectivity analysis of nodes over time, we find the central nodes according to many centrality definitions, we evaluate the set of active users and how this set evolves over time. We study the rich get richer phenomenon both from a connectivity point of view and from an economical richness (i.e. users balance) point of view. Finally we briefly present the same set of analysis for increasing amount thresholds instead of time.

The study of methods and tools for the analysis of complex networks has recently gained momentum, due to the presence of complex relational data in different fields. Network analysis has been applied in different scientific areas, like the analysis of biological systems [150], transportation systems [57] and social networks [128]. A novel application field is that of the networks modeling economic transactions occurring in some economic area. However, the analysis of real-life economy networks is not easy as there is no central entity registering all the transactions, since the transaction records are distributed over a large number of commercial entities or banks. An exception is that of transactions generated by digital cryptocurrencies, presented in Section 1. Current cryptocurrencies are based on a distributed public ledger to work, so providing a unique opportunity for analysis of currency transactions.

As we have seen in Section 1, the first true digital currency was Bitcoin, proposed in 2008 [195] by Satoshi Nakamoto, and the first client went online the 3rd of January 2009. From then, the system has gained wide mass media coverage and widespread popularity among the broad public of non specialists, resulting in the first example of cryptocurrency economy worthy of analysis. After nine years since the inception of Bitcoin, an economic community has risen around it. Bitcoin still represents a niche and peculiar economical community, nevertheless its importance in the real world has grown enough so that it no longer represents an experimental currency exploited only by computer science specialists. Several events of the Bitcoin economy, like the

wild speculation, the value fluctuation and a major exchange failure witness that a true economic system has born around it.

As explained in Section 1.2, the Bitcoin system operates according to a peer-to-peer philosophy, which avoids the need of a bank account maintained by a central authority. The transactions are grouped in blocks, recorded in a blockchain, and validated through a distributed consensus procedure. Therefore, Bitcoin keeps the entire transactions history public by design, so it represents one of the few economic communities that can be studied and analyzed in depth.

A distinctive characteristic of Bitcoin is that it puts together economic and technological aspects. This produces interesting interrelations between the Bitcoin real world economy and the technological aspects of the distributed protocol. These interrelations have influenced its development as much as its original design principles.

This chapter focuses on the analysis of the information obtainable from the blockchain, i.e. the definition and analysis of the users graph derived from the Bitcoin blockchain. The transactions encoded in the blockchain may be modeled by a multigraph, the transaction graph, whose nodes correspond to addresses, i.e. hash of public keys and edges correspond amount transfers between addresses. The users graph is derived from the transaction graph by a clustering process. Indeed, since each user may control several addresses, these can be grouped in a single cluster, by adopting some heuristics. Each cluster should ideally correspond to a single user. In this way, a new graph is defined, the users graph, whose nodes correspond to the users and whose edges correspond to value transfers between users.

Even if several works [230, 181, 201, 5, 156, 168] present analyses of the Bitcoin users graph to find out some interesting property of its economy, almost all of them consider an outdated state of the block chain, mainly an instance of the block-chain at the end of 2013. As shown in [54], the number of transactions has increased from roughly 10 millions in January 2013 to more than 100 millions of transactions in January 2016. This huge economic explosion, occurred in the last years, makes it interesting to analyse the features of the users graph with reference to a more recent state of the blockchain. We believe that an analysis of a recent log of the block-chain may enable a deeper understanding of the Bitcoin network and also highlight novel characteristics of the network arisen in the last few years. On the other hand, the analysis of such an huge amount of data, requires the definition of proper tools to perform both classical and more complex analyses. We first present the analysis assessing classical graph properties like densification, distance analysis, degree distribution, clustering coefficient, and several centrality measures. Then, we analyze properties strictly tied to the nature of Bitcoin, like rich-get-richer property, which measures the concentration of richness in the network.

2.1 Related Works

Several analyses of the Bitcoin network have been proposed. Most of them take in input the "user graph" that is extracted from the transactions graph through a well-established heuristic rule (see Section 1.5.2). This rule, already introduced in the seminal paper [195], and extensively described in [113], establishes that all the input addresses of a multi-input transaction belong to the same user (see Figure 2.1 for a simplified example of users graph). The rule is based on the observation that every input of a multi-input transaction must be signed with the right private key and this implies that the signer knows all the private keys of the transaction and so it is the owner of all the input addresses. The resulting graph approximates the real users graph, because the heuristics may underestimate or overestimate the common ownership of some addresses. While underestimation occurs because addresses of the same owner have not been used in the same transaction, overestimation may occur because a set of users may collectively sign the same transaction [235].

The heuristic rule has been subsequently used in most analysis, like in [230, 181, 201, 5, 156, 168]. Exceptions are [181] and [5], which also introduce a more sophisticated heuristic based on change addresses, i.e. the mechanism used to give money back to the input user in a transaction. [129] shows that the multi-input address heuristics generally exploited to derive the users graph is really effective and investigates the reason behind this observation. These are the high-levels of address reuse and avoidable merging; the existence of super-clusters with high centrality, and the incremental growth of address clusters. This encourages us to exploit this heuristics for the definition of the users graph, as shown in the following sections.

Let us now briefly review the most important analyses proposed. [230] considers only Bitcoin transactions carried out until May 2012. They discovered that the network contains a huge number of small transactions, but also a subset of transactions moving a large amount of money. The analyses are then focused on the large transactions in order to detect the ways amounts are accumulated and dispersed.

[156] does not apply any heuristic and directly analyses the transaction graph, extracted from the blockchain, whose state is considered as in May 2013. The authors identify an initial phase of growth of the Bitcoin network, characterized by a large fluctuation in the network characteristics and a trading phase characterized by more stable network measures. They find out that preferential attachment drives the growth of the network. The authors also analyse the Gini coefficient of the indegree distribution over time, doing something similar to what we do in Section 2.3.4, but the results are not comparable since their study is based on the simple transaction graph.

The main focus of the analyses presented in [181] is to highlight the gap between the potential and the actual anonymity of the Bitcoin protocol. The authors apply to the blockchain, as in April 2013, the two aforementioned heuristics to contract the transaction graph.

As most previous works, [168] considers the blockchain state at April 2013 and

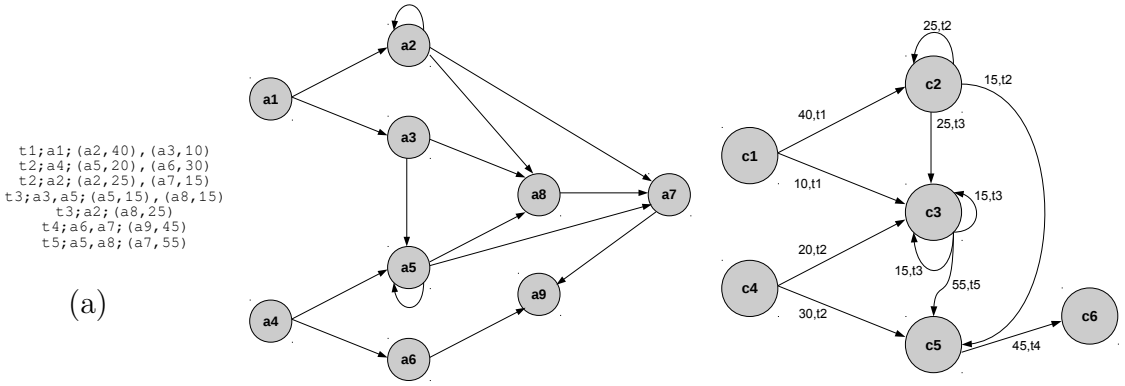


Figure 2.1: Simple example of users graph. (a) contains a list of simplified transactions, where each transaction is expressed in the format `timestamp ; comma separated list of input addresses ; comma separated list of couples ( output address , amount )`. (b) represents the corresponding transaction graph and (c) shows the derived users graph where cluster c3 contains the addresses a3, a5 and a8 and cluster c5 contains the addresses a6 and a7.

exploits only the first heuristic previously described for the contraction of the transaction graph. The authors categorize the transactions according to business categories by extracting the business tags of each address. Furthermore, they present an analysis of the geographic distribution of Bitcoin transactions.

The authors of [213] analyse the transaction networks of Bitcoin and Litecoin digital currencies. The analyses are applied to a graph where the nodes are the addresses of the Bitcoin users, while the edges represent a transaction between two addresses. Since no clustering is performed on the transaction graph, the results are characterized by a high level of approximation. For instance, since the top richest nodes detected in the network correspond to Bitcoin addresses, rather than users, it is possible that there exists a richer user which exploits different addresses to store its bitcoins.

2.2 Building the Users Graph

In this section, we present the clustering process which generates the users graph. In particular, we describe the clustering algorithm, some statistics on the cluster generated by the algorithm, and, finally, the data acquisition process we have performed.

2.2.1 The Clustering Algorithm

The Bitcoin dataset can be formally modelled by a weighted directed hypergraph $H = (A, T)$ where: A is the set of all addresses; T is the set of transactions, which can be modeled as a set of ordered pairs (A_1, A_2) with $A_1, A_2 \subseteq A$, meaning that the addresses in A_1 are paying the addresses in A_2 (see for instance [156]).

Moreover, to each transaction $s = (A_1, A_2) \in T$, we associate:

- a timestamp telling when the transaction took place.
- a distribution of amounts among the nodes in A_2 denoted as b_s. More formally, b_s is a function associating to each $a \in A_2$ a multiset of real numbers. Indeed, notice that there can be transactions associating to the same $a \in A_2$ more than one single amount.
- a fee ϕ_s (eventually 0) that associates to A_1 the voluntary taxes payed.

As seen in Section 1.2.1, in Bitcoin each user controls different pseudonymous addresses. In order to infer the users of the network, we want to cluster all the addresses managed by the same user so that each cluster will ideally correspond to a single user. In particular, we group addresses according to the following desired property.

Property 1 *For every two addresses x and y, if there exists a transaction (A_1, A_2) where $x, y \in A_1$, then x and y belong to a same cluster.*

Note that this is a sufficient but not necessary condition for x and y to be in the same cluster, since we merge clusters transitively: if for instance x and z belong to A_1 for some transaction $(A_1, A_2) \in T$ and z, y belong to A_3 for some transaction $(A_3, A_4) \in T$, then x, z, y will be assigned to the same cluster.

The clustering algorithm works as shown in procedure CLUSTER in Algorithm 1. Given the hypergraph $H = (A, T)$ above, the clustering algorithm produces a partition of A, i.e. the clusters $C_1, \ldots, C_k \subseteq A$ for some k, with $C_i \cap C_j = \emptyset$ $(1 \leq i, j \leq k$, $i \neq j)$ and $C_1 \cup \ldots \cup C_k = A$. We define the undirected graph G_H whose nodes are all the addresses in A and two nodes $x, y \in A$ are linked whether there exists a transaction $(A_1, A_2) \in T$ such that $x, y \in A_1$. Then the k connected components of G_A are our clusters $C_1, \ldots, C_k$.

The following result holds.

Lemma 1 *Given $H = (A, T)$, the clustering corresponding to the connected components $C_1, \ldots, C_k$ of G_H satisfies Property 1.*

It is worth observing that building G_H as described above can be costly: for each transaction $(A_1, A_2) \in T$, we have to create a clique among all the nodes in A_1, adding a quadratic number of edges, i.e. $|A_1| \cdot (|A_1| - 1)/2$. Instead of creating a clique, procedure CLUSTER in Algorithm 1 adds a simple path between the addresses

Algorithm 1: The graph building process

Input : A weighted directed hypergraph $H = (A, T)$, b_s for each $s \in T$
Output: A directed multigraph $G = (V, E, w)$

Procedure CLUSTER(H)
  $G'_H = (A, E') \leftarrow$ undirected graph with A as set of vertices and E' empty set of edges
  foreach $s = (A_1, A_2) \in T$ **do**
    Let $A_1 = \{a_1, a_2, \ldots, a_h\}$
    for $i \in \{1, \ldots, h-1\}$ **do** add $\{a_i, a_{i+1}\}$ to E'
  Let $C_1, \ldots, C_k$ be the connected components of G'_H
  return $C_1, \ldots, C_k$

$C_1, \ldots, C_k \leftarrow$ CLUSTER(H)
Let $c(a)$ be the vector associating to each $a \in A$ the cluster C_j such that $a \in C_j$
$\phi(C_i) \leftarrow 0$ for each C_i.
$G = (V, E, w) \leftarrow$ graph where $V = \{C_1, \ldots, C_k\}$ is the set of nodes, E is an empty set of arcs, $w : E \to \mathbb{R}$
foreach $s = (A_1, A_2) \in T$ **do**
  Let C_i be the unique cluster $c(a_1)$ for any $a_1 \in A_1$
  $\phi(C_i) \leftarrow \phi(C_i) + \phi_s$
  foreach $a_2 \in A_2$ **do**
    Let C_j be $c(a_2)$
    foreach $x \in b_s(a_2)$ **do**
      Add an arc e from C_i to C_j in E with weight $w(e)$ equal to x

in A_1, adding each time a linear number of edges. In other words, CLUSTER creates a graph G'_H whose set of nodes is A and whose set of edges is given by the following process: for each transaction $(A_1, A_2) \in T$, it creates a path among the nodes in A_1. The following property trivially holds.

Lemma 2 *G'_H and G_H have the same connected components.*

Once the clusters $V = \{C_1, \ldots, C_k\}$ have been identified in $H = (A, T)$ using G'_H, we create the weighted multigraph G whose set of nodes is V and there is an arc e from $C_i \in V$ to $C_j \in V$ whether there exists a transaction $(A_1, A_2) \in T$ such that $A_1 \cap C_i \neq \emptyset$ and $A_2 \cap C_j \neq \emptyset$. Roughly speaking, there is an arc from a cluster to another whether there exists a transaction from an address of the former to an address of the latter. Note that this is a multigraph since there can be several transactions from a cluster to another, possibly with different (or equal) amount.

Moreover, for each value $x \in b_s(a_2)$ with $a_2 \in A_2$, we create an arc with weight x, since, as explained before, a same transaction s can assign more than one amount to a vertex $a_2 \in A_2$. Finally, we define ϕ for each C_i as the sum of ϕ_s for each transaction s payed by C_i.

We will refer to G as Bitcoin users graph. The building method is summarized by Algorithm 1. It is worth noting that the whole process is linear in the size of H, i.e. $O(|A| + \sum_{(A_1,A_2)\in T}(|A_1| + |A_2|)$). An example of users graph is shown in Figure 2.1.

2.2.2 Clustering Statistics

For the sake of completeness, in Figure 2.2 we show the distribution of clusters size. It is worth observing, that this distribution follows a power law.

We report in Table 2.1 some basic statistics of our dataset, like the total number of addresses and the total number of transactions, and some statistics about the result of our clustering process.

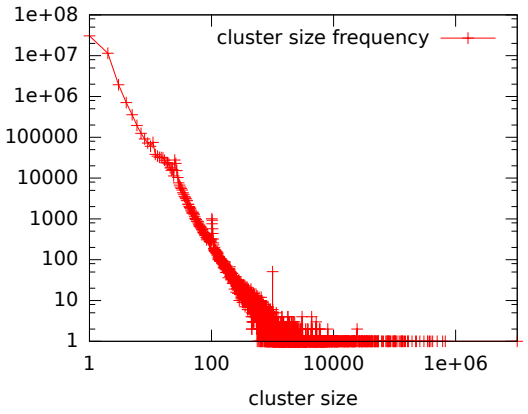


Figure 2.2: Distribution of Cluster Sizes.

In the lower part of Table 2.1 we report the list of the top ten biggest clusters we obtained. We observe that the size of these clusters is several order of magnitude bigger than the average size of the clusters, making the distribution of the clustering size heavy tailed (see Figure 2.2). We will see that several clusters in this top ten list are in the top ten list for other topological centrality measures.

2.2.3 Data acquisition

As we explained in Section 1.2 all the Bitcoin transaction history is publicly available in the blockchain as a countermeasure against double spending. To obtain the blockchain is sufficient to set up a Bitcoin node in the P2P network and start requesting blocks to the other nodes. This process can take a lot of time so, to avoid it, we

Number of Addresses, i.e. $\lvert A \rvert$	113 221 083
Number of Transactions, i.e. $\lvert T \rvert$	99 602 440
Number of Clusters, i.e. nodes of G	46 144 246
Number of Arcs of G	294 705 549

The 10 biggest clusters		
Cluster ID	Identity	Size
66 482	Mt. Gox	10 216 380
2 899 325	LocalBitcoins.com	676 402
26 784 111	GoCoin.com	611 885
11 032 019	AgoraMarket	497 995
12 388 597	EvolutionMarket	420 632
2 477 299	N/A	392 589
2 547 597	SilkRoadMarketplace	372 753
10 072 646	SilkRoad2Market	349 874
1 175 285	BTC-e.com1	348 438
11 828 673	999Dice.com	301 990

Table 2.1: Some Clustering Statistics.

used a blockchain already downloaded and stored in Protocol Buffers format [219]. We would like to thank Dr. Christian Decker and Prof. Roger Wattenhofer of the Distributed Computing Group, ETH Zurich for providing us the blockchain in such format. The blockchain used contains all the first 389 800 blocks, from the Genesis block until block height 389 799, hence containing all the Bitcoin transactions from 2009-01-03 18:15:05 GMT to 2015-12-23 09:40:52 GMT .

In Section 1.2.2 we have given a description of Bitcoin transactions and explained how they use scripts to work. We have decided to parse the blockchain only interpreting the `p2pkh`, `p2pk` and `p2sh` types of scripts, dropping the transaction outputs containing other kinds of scripts. We did this to not over-complicate our blockchain parser and because we thought that this kind of scripts where the ones used in transactions more suitable to apply our clustering heuristic, hence hoping to reduce the number of false positives returned by the heuristic clustering. In the end we successfully interpreted 295 144 677 scripts and failed to interpret 1 489 903 scripts, resulting in a coverage of 99.4977% of all transaction outputs. So we deem the information loss acceptable.

From the parsing of the raw blockchain with the script interpretation described before, we obtain our transactions dataset and on this dataset we apply our clustering algorithm and perform the analysis. We do not try to infer ourselves addresses identities and instead we rely on the public address tags datasets provided by [254, 108]. In the rest of this chapter to suppose a cluster identity we look for identity tags (provided by those services) associated to the addresses belonging to

that cluster.

2.3 Analysis and Results

In this section we study the topological properties of the Bitcoin users graph G built in Section 2.2. Recall that G is a weighted directed multigraph. We refer to U as the symmetric version of G, i.e. the graph where all the arcs become undirected.

In Section 2.3.1, we study the time evolution of G and U. For increasing values of time t we have considered just transactions that took place before t. We indicate with G^t (and U^t), with $1 \leq t \leq 20$, the graph induced by transactions whose time stamp is smaller than t, where t refers to the left part of Table 2.2. Analogously, we indicate with $\phi^t(u)$ the fees payed by u until time t, i.e. $\phi(u)$ induced by transactions with timestamp smaller than t. The timestamps chosen are at constant intervals in time but with an high initial offset. We chose to start the timestamp snapshots from the beginning of 2013 because we considered it the time when the Bitcoin economy started to rise significantly and was mature enough for a systematical analysis. Moreover, for each graph snapshot at each timestamp considered during the connectivity analysis phase in Section 2.3.1 we (non recursively) pruned the graph from the nodes with only one incoming arc. We choose to do so because otherwise the graph was biased from more recent nodes artificially isolated by the timestamp cutoff. Indeed, we noticed that most of those nodes corresponded to nodes that had just received a payment and didn't have enough time to use that value in a subsequent transaction. This pruning does not skew our analysis since the nodes pruned were nodes reached by only one incoming arc.

In Section 2.3.2 we report some centrality analysis based on the connectivity of the last snapshot of the network considering the degrees of the vertices, harmonic centrality and some spectral centrality [46].

In Section 2.3.3 we define active users and we study how the number of active users changes over time.

In Section 2.3.4, we define the *richness* of a node according to its number of incoming transactions and its balance. We study how the sets of richest nodes change over time, proving that richness tends to concentrate in terms of balance. We show that this holds even just considering active nodes.

In Section 2.3.5, we study the graph G for different transaction amounts. In particular, we call G_a (and U_a) the graph induced by transactions whose amount is smaller than a, with $1 \leq a \leq 12$, where the corresponding values of a are listed in the right part of Table 2.2. We remark that we have divided our time window in 20 equally spaced time stamps, each one corresponding to roughly 55 days.

Even though the nodes of G and U correspond to clusters of addresses as seen in Section 2.2, for the sake of simplicity, we will simply refer to them as vertices or nodes. Moreover, we will call the links in G as arcs, which are directed, and the ones of U as edges, which are instead undirected.

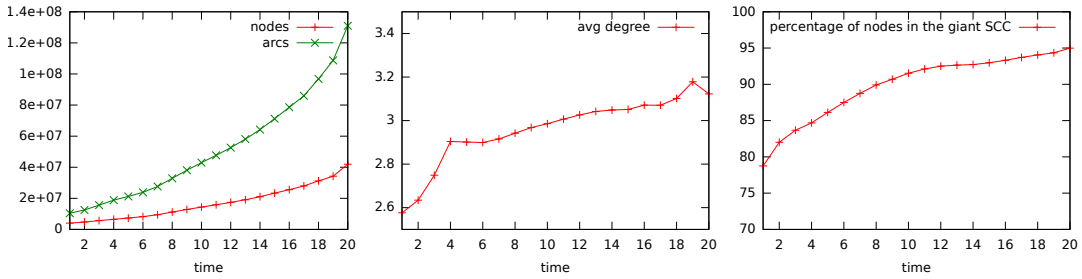
TIME t	SNAPSHOT
1	Tue Jan 01 00:00:00 GMT 2013
2	Sun Feb 24 07:41:02 GMT 2013
3	Fri Apr 19 15:22:04 GMT 2013
4	Wed Jun 12 23:03:06 GMT 2013
5	Tue Aug 06 06:44:08 GMT 2013
6	Sun Sep 29 14:25:10 GMT 2013
7	Fri Nov 22 22:06:12 GMT 2013
8	Thu Jan 16 05:47:14 GMT 2014
9	Tue Mar 11 13:28:16 GMT 2014
10	Sun May 04 21:09:18 GMT 2014
11	Sat Jun 28 04:50:20 GMT 2014
12	Thu Aug 21 12:31:22 GMT 2014
13	Tue Oct 14 20:12:24 GMT 2014
14	Mon Dec 08 03:53:26 GMT 2014
15	Sat Jan 31 11:34:28 GMT 2015
16	Thu Mar 26 19:15:30 GMT 2015
17	Wed May 20 02:56:32 GMT 2015
18	Mon Jul 13 10:37:34 GMT 2015
19	Sat Sep 05 18:18:36 GMT 2015
20	Wed Dec 23 9:40:52 GMT 2015

AMOUNT a	THRESHOLD
1	0.000 001 BTC
2	0.000 01 BTC
3	0.000 1 BTC
4	0.001 BTC
5	0.01 BTC
6	0.1 BTC
7	1 BTC
8	10 BTC
9	100 BTC
10	1 000 BTC
11	10 000 BTC
12	100 000 BTC

Table 2.2: The time series we considered (upper part) for G^t and U^t and the different amounts for G_a and U_a (lower part).

2.3.1 Connectivity Analysis Over Time

Since the interest of this section relies on the connectivity of the network, we consider G^t and U^t as simple graphs ignoring multiple arcs (or edges). Our analysis framework



(a) Number of nodes and arcs over time.

(b) Average out-degree over time.

(c) Percentage of nodes in the giant SCC over time.

Figure 2.3: Densification of G^t

have been implemented using WEBGRAPH [42].

Densification

This section aims to observe the densification process taking place in Bitcoin. This phenomenon is described by Figure 2.3.

- Figure 2.3(a) shows the increase of number of nodes and arcs over time in G^t. Since the time slots are equally spaced, the plot highlights that the increase of nodes and arcs is slightly more than linear.
- Figure 2.3(b) shows the increase of the average out-degree of the nodes in G^t for increasing values of t (note that the average out-degree and the average in-degree are the same). This plot highlights that the number of arcs in G^t increases faster than the number of nodes.
- Figure 2.3(c) shows the behaviour of the percentage of nodes in the giant strongly connected component of G^t with respect to the total number of vertices in G^t, i.e. $|V^t|$. This value quickly increases, meaning that, even though $|V^t|$ grows quite fast (Figure 2.3(a)), the number of nodes in the giant strongly connected component grows much faster making the network much more robust.

Distance Analysis

To perform a distance analysis we have computed diameter and average distance of the Bitcoin network. The distance $d(u, v)$ from a node u to a node v in a graph is the length of the shortest path from u to v. The diameter is defined as the $\max_{(u,v)\in V^t\times V^t} d(u, v)$, while the average distance is simply $\frac{1}{|V^t|^2}\sum_{(u,v)\in V^t\times V^t} d(u, v)$. In order to get more robust analysis, we have done these measurements considering the giant connected component of U^t for increasing values of t, where two users

are connected whether they exchanged Bitcoins, ignoring the direction of the link. The average distance has been approximated using [43], while the diameter has been computed exactly using [48]. Note that, for a graph of n nodes and m edges, computing the average distance and the diameter requires $O(n \cdot m)$. The algorithm in [43] allows to approximate the average distance in $O(m)$ and the algorithm in [48] allows to compute exactly the diameter in $O(m)$ in practice in real world graphs.

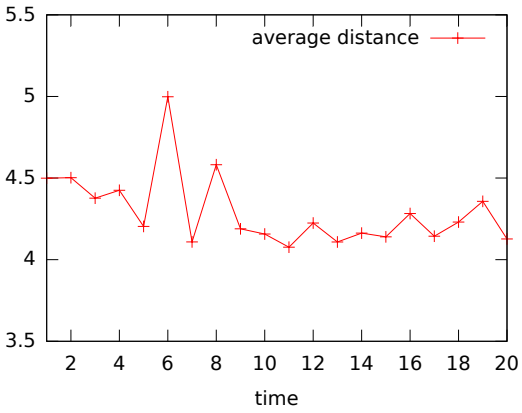


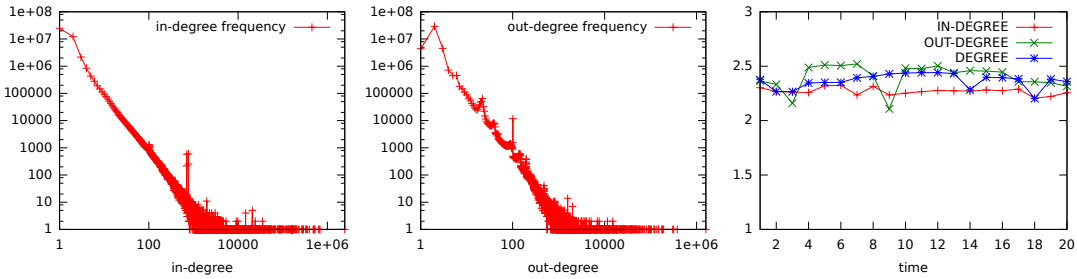
Figure 2.4: Average distance of U^t for increasing values of t.

As observed for many other real world networks [166], we have seen that the diameter is not increasing. Surprisingly, the diameter is constant and very long (i.e. 2050) if compared to the diameter of many other real-world networks, like Facebook [11], where the number of vertices is much higher and the diameter is 41, and others (see `lasagne-unifi.sourceforge.net`). Our preliminary observations suggest that this peculiarity of the Bitcoin users graph is caused by the fact that transactions are also used to merge and split user funds and not just for payments, as explained in Section 1.2.2. This is consistent with rare user cases observed in the past, obfuscating funds ownership using long fund splits chains (as noted for example in [230]). We plan to investigate extensively this topic in our future works.

Figure 2.4 shows the slow decrease over time of the low average distance. Note that this small average value compared to the high value of the diameter highlights that the nodes connected by long paths are present but few.

Degree Distribution

In Figure 2.5(a) and Figure 2.5(b) we show respectively the in-degree and the out-degree distributions of G^t with $t = 20$. As a further remark, we have seen that also the degree distribution of U^t, which for brevity is not shown here, follows a similar behaviour. It can be noticed that in both the plots (a) and (b) there are some outliers: there are some spikes close to $x = 1000$ in Figure 2.5(a), and to $x = 100$ in Figure 2.5(b). These spikes have been object of preliminary investigations in [173].

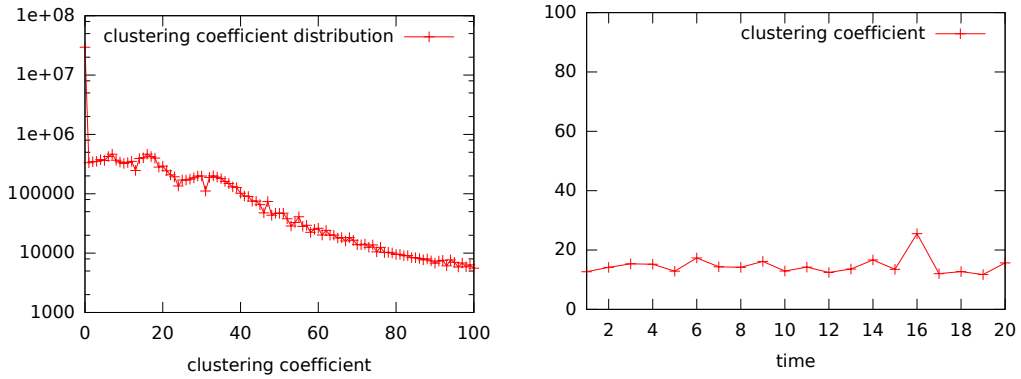


(a) In-degree distribution of G^t with $t = 20$.

(b) Out-degree distribution of G^t with $t = 20$

(c) Power law exponent as the time goes by.

Figure 2.5: Behaviour of Degree Distributions



(a) Clustering Coefficient of U^t with $t = 20$.

(b) Average Clustering Coefficient at passing time.

Figure 2.6: Behaviour of Clustering Coefficient

These have been shown to be related to particular topological patterns which are more likely due to transactions caused by unexpected users behavior rather than normal economic interaction.

All the distributions above follow a power law. In Figure 2.5(c) we show the power law exponent for increasing values of t for the in-degree distribution of G^t (red line), the out-degree distribution of G^t (green line), and the degree distribution of U^t (blue line). The power law exponent seems to be constant over time confirming the estimations done in [156].

Clustering Coefficient

The clustering coefficient $c(v)$ of a node u in $U^t = (V^t, E^t)$ is defined as the percentage of the number of triangles involving v with respect to the possible number of triangles that could involve v. More formally, $c(v)$ is $\frac{2|\{(w,z)\in E^t:\ w,z\in N(v), w\neq z\}|}{|N(v)|\cdot(|N(v)|-1)} \cdot 100$, where $N(v)$ denotes the set of vertices which are neighbors of v in U^t. Since com-

	DEGREE		IN-DEGREE		OUT-DEGREE	
	IDENTITY	VALUE	IDENTITY	VALUE	IDENTITY	VALUE
1	Mt. Gox	3 386 581	Mt. Gox	2 452 049	Mt. Gox	1 591 319
2	LocalBitcoins.com	902 151	BTC-e.com1	683 875	2477299	381 426
3	2477299	848 176	LocalBitcoins.com	650 269	SatoshiDice.com	317 742
4	BTC-e.com1	740 402	AgoraMarket	636 969	LocalBitcoins.com	301 692
5	AgoraMarket	722 331	SilkRoadMarketplace	527 718	14782788	191 867
6	SilkRoadMarketplace	577 124	2477299	511 239	MoonBit.co.in	180 161
7	BitPay.com1	500 990	BitPay.com1	493 067	FaucetBOX.com	178 349
8	BTC-e.com2	492 219	BTC-e.com2	479 452	26638073	176 508
9	Cryptsy.com	461 111	BitPay.com2	394 447	Cryptsy.com	148 015
10	BitPay.com2	401 254	Cryptsy.com	361 298	23144512	146 624

Table 2.3: The top 10 central nodes according to degree centralities: degree in U^t, in-degree and out-degree in G^t for the last snapshot, i.e. $t = 20$. For the unknown identity we report the number of the identifier in our dataset.

	HARMONIC		PAGE-RANK		EIGENVECTOR	
	IDENTITY	VALUE	IDENTITY	VALUE	IDENTITY	VALUE
1	Mt. Gox	11 798 171	Mt. Gox	0.0183119	Mt. Gox	0.5369751
2	2477299	10 447 302	BTC-e.com1	0.0059199	SilkRoadMarketplace	0.0379121
3	LocalBitcoins.com	10 320 862	BTC-e.com2	0.0057736	BTC-e.com2	0.0303681
4	Cex.io	10 144 968	LocalBitcoins.com	0.0053508	BitPay.com1	0.0282049
5	FaucetBOX.com	10 136 604	SilkRoadMarketplace	0.0046309	Cex.io	0.0254740
6	26638073	10 071 881	AgoraMarket	0.0042383	SatoshiDice.com	0.0247234
7	MoonBit.co.in	10 065 853	BitPay.com1	0.0041264	LocalBitcoins.com	0.0242701
8	19860816	10 025 701	BitPay.com2	0.0036458	BTC-e.com1	0.0202184
9	Poloniex.com	9 976 766	SatoshiDice.com	0.0031628	BitPay.com2	0.0185086
10	Bittrex.com	9 926 321	2477299	0.0026843	AgoraMarket	0.0171055

Table 2.4: The top 10 central nodes according to harmonic, page-rank, and eigenvector centrality in U^t, for the last snapshot, i.e. $t = 20$.

puting exactly the clustering coefficients of a graph with m edges is costly, i.e. the best-known algorithm for computing triangles takes $O(m^{3/2})$ [147, 58], we have used approximation algorithms based on min-sketches. In particular, we have used [18], setting the size of the sketches equal to 64. Using the latter algorithm, we have computed the average $c(v)$ for each graph U^t for increasing values of t, using the definition of global clustering coefficient by Watts and Strogatz [273]. In Figure 2.6(b) we report these values showing that the average clustering coefficient is basically constant over time. The order of magnitude of the correspondent values is similar to the one of the values reported in [2] for other complex networks. In Figure 2.6(a) we report also the distribution of $c(v)$ in U^t with $t = 20$. This distributions shows that the great majority of the vertices have clustering coefficient equal to 0. Among them, there are vertices composing long paths without shortcuts: these are the main responsible of the long diameter we have observed in Section 2.3.1.

Finally, we remark that, despite the high diameter, the slow decrease over time

of the low average distance (shown in Figure 2.4), together with the relatively high clustering coefficient suggests a small world phenomenon.

2.3.2 Centrality Analysis

In this section we report the most central vertices in the Bitcoin network. These vertices correspond to the most active vertices in G^t or the ones that play a crucial role for the connectivity of the network (see [46], for more details about centrality measures). Given $G^t = (V^t, E^t, w^t)$, the centrality of u can be defined as follows.

- DEGREE: That is the degree of u in U^t, i.e. the number of nodes paying or payed by u. Central nodes are supposed to have many connections.
- IN-DEGREE: That is the in-degree of u in G^t, i.e. the number of nodes that payed u.
- OUT-DEGREE: That is the out-degree of u in G^t, that corresponds to the number of nodes which have been payed by u.
- HARMONIC: That is $\sum_{v \in V^t} \frac{1}{d(u,v)}$, where $d(u, v)$ is the distance between u and v in G^t. According to this measure, a node is central whether its distance from the others is small. This centrality is basically a variant of the closeness centrality which takes into account the disconnection of the network. A large value means high centrality [46].
- PAGE-RANK: this measure is popular because of its usage in Googles ranking algorithm [205]. The score of a node corresponds to the probability of visiting that node when performing a random walk in the graph or of visiting the node intentionally.
- EIGENVECTOR: this spectral measure uses the left dominant eigenvector of the plain adjacency matrix. Spectral measures, like also PAGE-RANK, in general are based on the assumption that a node is important if it is linked to by other important nodes. In this case, the centrality of a node is the result of a convergent iterative process which replaces the score of a node with the sum of the scores of its predecessors [19].

In Table 2.3, we report the top-k users according to the above measures based on degree, with $k = 10$. On the other hand, Table 2.4 shows the top-k with $k = 10$ according to the other measures. The HARMONIC centrality has been computed using [45] while the PAGE-RANK and EIGENVECTOR centrality have been approximated using the software provided by [46].

We remark there is a big correlations between degree centralities and the other measures. As expected, the selected central vertices are almost all very popular. Mt. Gox (the most famous Bitcoin exchange before its failure at the beginning

of 2014) is the most central according to all the measures not only considering its local connectivity, i.e. the degrees, but also for the connectivity of the whole network. The same applies to LocalBitcoins.com (the largest P2P Bitcoin trading platform). Interestingly, it seems difficult to identify some users, as for instance the one corresponding to vertex 2477299, that seems to be very central.

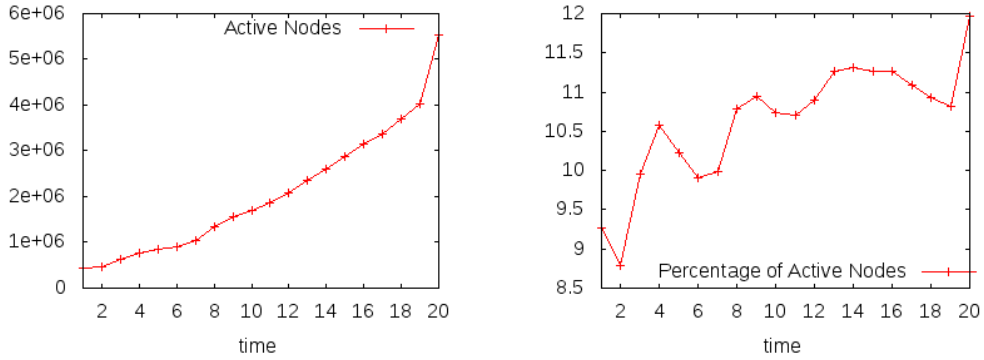
The degree centralities consider the size of the local community of each node, so that the vertices shown in Table 2.3 are the ones which have participated to (respectively, received or send) more transactions. Some popular nodes are equally central both considering incoming and outgoing transactions. On the other hand, some nodes, like MoonBit.co.in or FaucetBOX.com, are more likely paying than receiving with respect to others, while some others, like AgoraMarket and SilkRoadMarketplace are more likely receiving than paying. This is consistent with the nodes category since the first two are faucets (giving away tiny amounts of BTC for free), while the second ones are marketplaces accepting payments in BTC.

The HARMONIC rank highlights which nodes are closer to all the others from a global point of view considering shortest paths, while PAGE-RANK and EIGENVECTOR consider arbitrary paths. The nodes in the first column of Table 2.4 are nodes basically making shortcuts between different part of the network. It is interesting to note that the nodes ranked from 6 to 10 in the first column do not appear in the rightmost columns. We argue that considering shortest paths rather than arbitrary paths significantly changes the centrality of a node probably because of the several alternative paths that are present in the network. On the other hand, as expected, the rightmost columns of Table 2.4, both concerning spectral measures, are quite similar. These nodes are the ones more likely to meet when performing a random walk, which can obviously take place along arbitrary paths. However, we notice a particular similarity with DEGREE and IN-DEGREE centrality.

2.3.3 Active Users Over Time

We've shown in Section 2.2.3 that transactions are divided among standard and non-standard, where non-standard transactions are not relayed on the network nor included in blocks by compliant nodes. We remark that non-standard transactions can still be valid so that they are accepted if they appear in the blockchain. We've shown how scripts can make transactions non-standard but other factors have a role as well. The rules defining standard transactions are dictated and updated over time by the official client implementation. Regarding the possible amount spendable by transactions, the standardization rules introduced the concept of dust [68]: *an amount* a *is called* dust *if creating a transaction to spend* a *would cost more in recommended fees than* a. The rationale behind the dust definition is that the amount transferred in the transaction is economically unreasonable to be spent.

So we consider the *dust limit* as the threshold to consider a balance negligible. This is because in the majority compliant network would be almost impossible for a user with a current balance lower than the dust limit to create a transaction that will



(a) Active nodes at passing time.

(b) Percentage of active nodes at passing time.

Figure 2.7: Active nodes

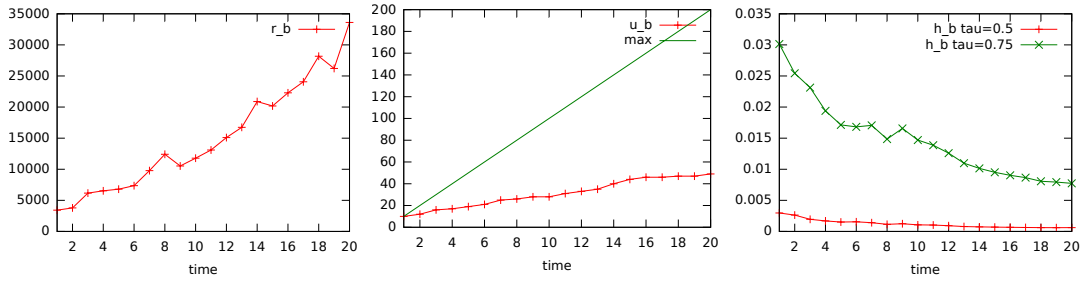
be actually included in the blockchain. It is worth remarking that the value stored by such a user is not lost. The user cannot spend that value alone in a transaction but he can cooperate with others to collect the dust to obtain transactions with total value high enough to be considered standard. There exist protocol to do so (called *dust collectors*) to collect the dust and give it directly to the miners. What we want to state is that the economic power of a below dust account is irrelevant, in the sense that it cannot be spent by the user alone to benefit him (as regular value would instead do). As a result, in practice we can consider all users with a current balance value lower than the dust limit equivalent to users with a zero balance. The actual value of the dust limit has varied over time and has been dependent of other parameters (such as the *minimum relay transaction fee* value) that have varied over time as well or have been left under discretion of nodes owners. The current standard can be seen in [68]. For the sake of simplicity, in order to set a fixed dust limit over the entire time span of our analysis, we've decided to choose as dust limit the most conservative "official" value (i.e., the lowest) adopted during the same time span, which is 0.00000546 BTC.

We consider each cluster as *active* at a give time if its current balance is greater than the dust limit.

In Figure 2.7(a), we show how the number of active nodes changes over time. The increase seems to be slightly more than linear. For $t = 20$, we think the estimation does not follow the trend because of the clustering phase, which is less effective to recognize very recent users, as shown in Section 2.2. This plot should be compared with the one in Figure 2.3(a), which showed the increase of nodes over time. In Figure 2.7(b), we show the percentage of active nodes with respect to the total number of nodes in the network over time. This ratio slightly increases, passing from 9% to 12%, suggesting that a greater portion of the network is remaining active with the passing of time.

RANK	IN-TRANS		BALANCE	
	IDENTITY	VALUE	IDENTITY	VALUE
1	Mt. Gox	22 399 043	39912924	169 731
2	SatoshiDice.com	12 879 343	23638585	157 997
3	LuckyB.it	3 620 428	542746	87 111
4	BitZillions.com	1 651 456	Mt. Gox	81 492
5	BTC-e.com1	1 430 992	177808	79 957
6	LocalBitcoins.com	1 356 029	6128144	69 370
7	Xapo.com	1 264 648	10597666	66 650
8	AgoraMarket	1 184 011	10467915	66 612
9	BetcoinDice.tm	1 159 024	10484095	66 583
10	2477299	1 101 024	10475912	66 452

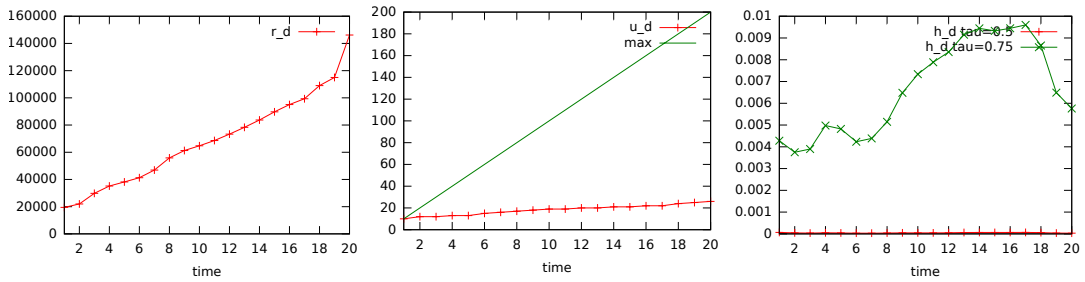
Table 2.5: The top 10 richest nodes in G^t with $t = 20$.



(a) r_b^t over time

(b) u_b^t over time and the maximum u_b^t possible

(c) h_b^t over time, fixing $\tau = 0.5$ and $\tau = 0.75$.



(d) r_d^t over time

(e) u_d^t over time and the maximum u_d^t possible

(f) h_d^t over time, fixing $\tau = 0.5$ and $\tau = 0.75$.

Figure 2.8: Verifying Property 2.1, 2.2, and 2.3 for the definitions of richness given by Equation 2.1 (upper part, respectively (a), (b), and (c)) and Equation 2.2 (lower part, respectively (d), (e), and (f)).

2.3.4 Rich get Richer and Concentration of Richness

This section is devoted to verify the *rich get richer* hypothesis and measure the concentration of richness, both on the balance and the connectivity point of view. Indeed, we consider two different definitions of richness: we say that a user is rich whether its balance or its number of incoming transactions is high with respect to the other users in the network.

We remark that our observations are not an artifact of the protocol but rely on truly economical and connectivity properties. Indeed, they depend only on the number of transactions and on their amount, that only relate to users behaviour.

We have restricted our studies to active users only. Indeed, from a balance point of view, considering only active users does not change the richness ordering. It only ignores the long tail of lower clusters with zero or negligible current balances, hence influencing the overall average but not the ordering. In such a context, we show the rich get richer phenomenon and an high concentration of richness even neglecting very poor users, i.e. non-active users. Hence, if we compare our results with the ones in [77] we can see how despite restricting just to active users, the trend observable in the graphs is the same.

Thus, in the following, we will consider G^t as the graph without non-active nodes.

We aim to verify the following properties for both the definitions.

Property 2

1. *The richest users at time t are richer than the richest users at time $t' < t$.*
2. *The richest users at a certain time t tend to remain the richest at time $t' > t$.*
3. *The richness gets more concentrated with the progression of time.*

Given the weighted multigraph $G^t = (V^t, E^t, w^t)$ (without non-active nodes) and ϕ^t for each $v \in V^t$, we formally define the richness of a node $u \in V^t$ as its balance $b^t(u)$ or its number of incoming transactions $d_t(u)$ as follows.

$$b^t(u) = \sum_{(v,u)\in E^t} w(v,u) - \sum_{(u,v)\in E^t} w(u,v) - \phi^t(u) + \beta^t(u) \tag{2.1}$$

$$d^t(u) = |\{(v,u) \,:\, (v,u) \in E^t\}| \tag{2.2}$$

Observe that the measure $b^t(u)$ is taking into account also the fees payed by user u as $\phi^t(u)$. Moreover, $\beta^t(u)$ is the increase of the balance of u up to time t (eventually 0) coming from special transactions called *coinbase* whose aim is minting new coins (as explained in Section 1.2.3).

Given an integer k, we denote respectively as B_k^t and D_k^t the k nodes having maximum b^t and d^t. Table 2.5 shows the sets D_k^t (IN-TRANS columns) and B_k^t

(BALANCE columns) for $t = 20$ and $k = 10$. The identity of these nodes is the name or the identifier of the node in our dataset in the case we do not know its name. As far as we know, many of the BALANCE column identifiers correspond to Bitcoin accumulator addresses (single addresses that received huge amounts of BTCs over time without spending them).

Note that, the measure $d^t(u)$ corresponds to the in-degree of u in the multigraph, i.e. the number of transaction outputs paying u. This is different from the in-degree considered in Section 2.3.2 which corresponds to the degree in the simple graph, i.e. the number of users paying u.

A definition similar to $d^t(u)$ can be done considering the transactions outgoing from u, obtaining similar results. However, the number of transactions outgoing from a user can be arbitrarily increased by the user (performing transactions with small amounts) to increase its connectivity. We argue that our definition of $d^t(u)$ is more robust for modelling economic importance.

Verifying Property 2.1

To verify that the richest users in G^t are richer than the richest in $G^{t'}$ with $t' < t$, we study the following quantities over time.

$$r_b^t = \frac{\sum_{u \in B_k^t} b^t(u)/k}{\sum_{u \in V^t} b^t(u)/|V^t|}, \qquad r_d^t = \frac{\sum_{u \in D_k^t} d^t(u)/k}{\sum_{u \in V^t} d^t(u)/|V^t|}$$

Basically, r_b^t (resp. r_d^t) is the ratio between the average balance (resp. incoming transactions count) of the top-k richest users with respect to the average balance (resp. incoming transactions count) of all the users in the network. As this ratio gets higher, the disparity of the richest nodes with respect to all the others gets bigger.

Figure 2.8(a) clearly shows that r_b^t increases over time, meaning that this disparity increases. On the other hand, Figure 2.8(d) shows that the same applies to r_d^t.

Verifying Property 2.2

In order to test the diversity of the richest node sets, i.e. B_k^t (resp. D_K^t), varying t, we study the following quantities.

$$u_b^t = |\bigcup_{i=1}^{t} B_k^i| \qquad u_d^t = |\bigcup_{i=1}^{t} D_k^i|$$

Since $|B_k^t| = k$ (resp. $|D_k^t| = k$), in the case the richest node sets does not change for each time i with $1 \leq i \leq t$, we have $u_b^t = k$ (resp. $u_d^t = k$). On the other hand, if the sets B_k^i (resp. D_k^i) change completely for each i then we have $u_b^t = t \cdot k$ (resp. $u_d^t = t \cdot k$).

Figure 2.8(b) and (e) show the behavior of u_b^t and u_d^t over time with respect to the expected behaviour if sets would change (green line). Both the sets B_k^t and D_k^t are very stable. Fixing $t = 20$ and $k = 10$, u_b^t, the set of all the k-richest nodes in the *history* of Bitcoin, is less than 50 instead of 200. This stability seems to be even more evident in the case of u_d^t, where $|\bigcup_{i=1}^{t} D_k^i|$ is smaller than 30 instead of 200.

Verifying Property 2.3

To measure the concentration of richness in $G^t = (V^t, E^t, w^t)$, fixing a threshold τ, we considered the following measures.

$$h_b^t = \min \left\{ k \ : \ \frac{\sum_{u \in B_k^t} b^t(u)}{\sum_{u \in V^t} b^t(u)} > \tau \right\} / |V^t|$$

$$h_d^t = \min \left\{ k \ : \ \frac{\sum_{u \in D_k^t} d^t(u)}{\sum_{u \in V^t} d^t(u)} > \tau \right\} / |V^t|$$

As an example, consider $\tau = 0.75$:

- h_b^t is the minimum (normalized) k such that B_k^t owns the 75% of the richness, in terms of balance, of the whole network;
- h_d^t is the minimum (normalized) k such that D_k^t owns the 75% of incoming connections of the network.

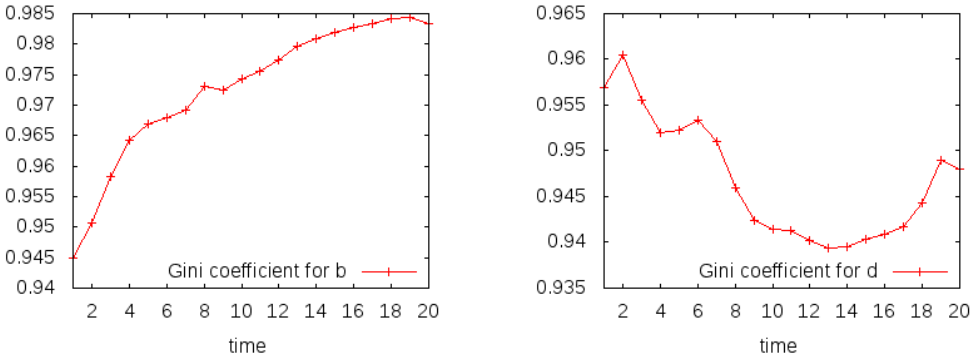
A small value for h_b^t (or h_d^t) means that the richness is concentrated in few users. The increase of concentration of richness can be witnessed checking whether h_b^t (and h_d^t) decreases over time.

In Figure 2.8(c) and (f), we report the behaviour of both h_b^t and h_d^t for increasing values of time t in our time series setting $\tau = 0.5$ and $\tau = 0.75$. Figure 2.8(c) refers to h_b^t and clearly decreases over time showing that the balance becomes more concentrated as the time passes; this is more evident especially considering an higher value of τ, i.e. $\tau = 0.75$.

On the other hand, the results for h_d^t, reported in Figure 2.8(f), shows that connectivity does satisfy Property 2.3. Indeed, contrarily to h_b^t, h_d^t seems to increase over time (except for the last data points). We argue that, for the densification process shown in Section 2.3.1, the increase of arcs is too big to be suitably absorbed from a same percentage of nodes.

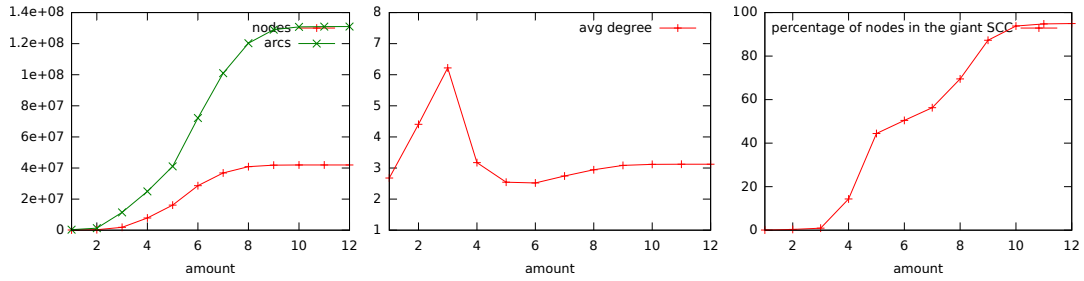
Gini Coefficient

As a further measure to support our concentration studies we have also considered the Gini coefficient, a well established inequality measure defined as follows: given



(a) Gini coefficient for b over time. (b) Gini coefficient for d over time.

Figure 2.9: Gini coefficient for both the definitions of richness.



(a) Number of nodes and arcs of G_a for increasing amounts a.

(b) Average degree in G_a for increasing amounts a.

(c) Percentage of nodes in the giant SCC of G_a for increasing amounts a.

Figure 2.10: Densification of G_a for increasing amounts a (see also Table 2.2)

x_i indexed in non-decreasing order, the Gini coefficient is

$$g = \frac{2\sum_{i=1}^{n} i x_i}{n\sum_{i=1}^{n} x_i} - \frac{n+1}{n}$$

In our case, x_i is the richness of the i-th node u in the network G^t, respectively defined as $b^t(u)$ and $d^t(u)$. In Figure 2.9, we show how the Gini coefficient changes over time, considering both $b^t(u)$ and $d^t(u)$, respectively in Figure 2.9(a) and Figure 2.9(b). First of all, notice that the values are very high in general and relatively close to the maximum possible, i.e. 1. In the first case, when richness is defined as balance, the Gini coefficient clearly increases over time, confirming the fact that the balance based definition richness satisfies Property 2. On the other hand, in the case of Figure 2.9(b), when richness is defined in terms of connectivity, the Gini coefficient decreases over time. This fact is consistent with our findings in the previous sections, since we have seen that connectivity richness does not satisfy Property 2.3.

2.3.5 Further Analysis for Different Transaction Amounts

In this section we show the growth of graph G_a for increasing values of a. Recall that G_a is the graph G built in Section 2.2 induced by transactions whose amount is smaller than a, where the correspondence between a and real BTCs is provided by Table 2.2.

Figure 2.10(a) shows the increase of number of nodes and arcs in G_a for increasing a. We can see that the bigger increase of nodes is when transactions of amount between 3 and 7 are introduced (see the angles of red and green lines). From $a \geq 8$, the number of nodes and arcs is stable, meaning that not many transactions have an amount greater than 1 BTC. Figure 2.10(b) shows the average degree for increasing a. There is a spike for $a = 3$ meaning that the maximum relative increase of edges with respect to the nodes is when introducing transaction with amount between 0.00001 BTC and 0.0001 BTC. This is due to the fact that nodes doing this kind of transactions often perform even smaller transactions. Looking at Figure 2.10(c), we can see that these smaller transactions are not well connected meaning that these take place in independent parts of the network: some parts of the network are giving, some other parts are receiving, but rarely they are exchanging. This suggests that no well connected micro-economy is present, but all the transactions up to $a = 8$ are needed to make the great majority of the network strongly connected. Interestingly, we can see a similar shape between Figure 2.10(c) and the green line in Figure 2.10(a).

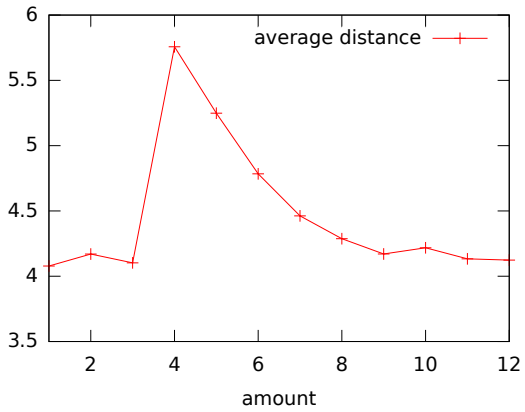


Figure 2.11: Average distance of U_a for increasing values of a.

Figure 2.11 shows the average distance in U_a, the undirected version of G_a. The starting increase for $a < 4$ is due to the fact that we are merging independent parts of the network, creating new paths (this is consistent with Figure 2.10(c)). The decrease for $a > 4$ is due to the fact that bigger transactions are creating shortcuts for relationships already existing.

For the sake of completeness, we mention that we observed a stable power law exponent for increasing values of a for all the degree distributions, similarly to

Figure 2.5. In particular, the exponent is constantly about 2.3 for the out-degree distribution in G_a and the degree distribution in U_a; it is 2.2 for the in-degree distribution in G_a.

3

Detecting Artificial Behaviours

Abstract

The analysis performed in the previous chapter let us find out the presence of some unexpected results. In particular we noticed an high constant diameter, despite a low average distance between nodes, and the presence of evident outliers in the indegree distribution. In this chapter we isolated the users and transactions responsible of such results. We explain how a single transaction pattern, named GPS*-chain, adopted by a minority of transactions is responsible for both these two macroscopic effects on the graph. We try to give an explanation to why this peculiar transaction patterns are used by users, concluding that they are probably result of an artificial user behaviours instead of being an effect of real users economic interaction.*

In the previous chapter we have analysed several properties of the Bitcoin users graph. In particular, we have shown that the graph presents many feature characteristics of the small-world phenomenon, but also some odd behaviours. As a matter of fact, while the average distance between nodes is low, the graph presents a high value of the diameter. This highlights the presence of few pairs of nodes connected by long paths only. Furthermore, the in-degree frequency distribution of the nodes presents some relevant outliers. As power law degree sequences and small diameter are well-established requirements [90] for complex networks in general to assess the small-world and scale-free phenomena, we argue that it is interesting to study these anomalies of our network and the reasons behind them. A possible explanation is that the odd behaviors are caused by users exploiting Bitcoin not only for ordinary transactions, but rather for other activities, like fund management and, possibly, attacks, as stated in our following Conjecture.

Conjecture 1 *(a) The indegree frequency distribution anomalies and (b) the high diameter are caused by uncommon artificial users behavior*[1] *rather than being inherent properties of the system.*

[1] I.e. not representing a value exchange between users.

This chapter supports this conjecture and our analysis show that these behaviours are a consequence of particular anomalous chains of transactions, which we call suspicious transactions. A (α, k)-suspicious transaction is such that the only input address pays all the output addresses, which are at least k, α BTC except at most one, which can be the recipient of an arbitrary amount. We show that special combinations of suspicious transactions, called pseudo-spam transactions, are the sole cause of indegree frequency distribution anomalies and can be further agglomerated in just one cluster each, lowering the diameter length and hence obtaining a much shorter diameter. We remark how these so called suspicious transactions are not "suspicious" by themselves, i.e. they might very well be the result of normal users activity. They were labeled as such because they appeared during both our manual observations presented in Section 3.2 and because they become interesting when combined together in long chains.

3.1 Related Work

The novelty of the users graph derived from the Bitcoin blockchain is that nodes represent pseudonymous users and a link between nodes represents an economic interaction, i.e. a flow of value between nodes. Interesting analyses of the Bitcoin ecosystem start from an analysis of this graph. Further interesting findings may be obtained through an integration of this analysis with that of further social networks [120]. For instance, analyzing the sentiment of opinions and information distributed on Twitter regarding Bitcoin and comparing with Bitcoins price, can be used to check if there is a correlation between Twitter sentiment and BTC price fluctuation. Other works have pointed out the strong correlation between Bitcoins price and the search of the "bitcoin" term, calculated by Google Trends [176].

Given the huge amount of public transaction data stored in the blockchain it is not surprising that some studies, aimed at extracting meaningful information from it, have been conducted in the past. Mostly though such studies were side effects of broader deanonymization attempts and almost always relied on human driven manual or automatic inspection. An example is [231], where the authors try to manually track a precise user. Another example can be found in [181] where the authors use the results of their presented deanonymization attack to manually follow transaction chains believed to be part of illicit activities such as thefts. Other results have been obtained in the literature through human observation made possible by blockchain data visualization tools. In [179] the authors present such a visualization tool developed to inspect the Bitcoin blockchain. The visual representation of the chaotic transaction data has the advantage of allowing a fast human recognition of special patterns, but it is not designed to detect interesting patterns in an automatic manner. Another visualization tool is presented in [76], and used by the authors to analyse the transactions created by mixing services found in [191]. Finally some interesting generic transaction patterns are shown in [230], but they

are limited to the use case considered and their impact on the entire users graph is not studied. We believe our work to be novel since it is the first attempt we know of trying to automatically characterize and detect particular transaction patterns in the blockchain.

3.2 Our Observations

In this section we present our observations concerning the two main anomalies we want to study: outliers in the indegree frequency distribution and the surprisingly high diameter. Based on the insight obtained from such observations we will give a formalization of interesting transaction chains (and the transactions they are made of) in Section 3.3. We will then show in Section 3.5 how these chains are responsible for the anomalies we observed.

3.2.1 Indegree Frequency Distribution

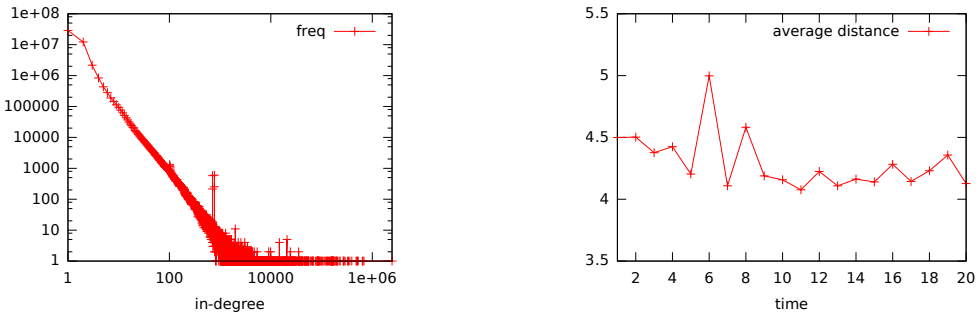
As observed in the previous chapter, the indegree frequency distribution of the nodes of the users graph follows a power law, as expected in a complex network. For the sake of completeness we report this indegree frequency distribution in Figure 3.1(a) (note the log-log scale). In the previous chapter we also observed that the exponent of this frequency distribution is stable over time. (i.e. the exponent for the frequency distribution of the indegrees of the users graph obtained from the blockchain at discrete time intervals).[2] However, some particular values for the indegrees are worth studying as shown next.

Looking at Figure 3.1(a), it strikes the eye the presence of some evident outliers. It is interesting to understand what is the reason behind such an unexpected number of users receiving a certain number of payments and whether this could be explained as a macro effect of some interesting user behaviours.

In order to isolate the indegree outliers to be analysed, we consider all the degrees having the following property (with 10 as arbitrary chosen value for the parameter k).

Definition 1 *Let $y = I(x)$ be the indegree frequency distribution, where $I(x)$ is the number of nodes of the users graph with indegree x, and let Δ be the maximum indegree in the graph and $k \in \mathbb{N}$ a parameter. For every $k < x \leq \Delta$, x is a* suspicious outlier *if $I(x)$ is at least one order of magnitude greater than the average $I(x-i)$ with $1 \leq i \leq k$, i.e. if $I(x) > 10 \cdot \frac{\sum_{i=1}^{k} I(x-i)}{k}$.*

[2]We remark that we are considering the Bitcoin users graph (see Section 3.1), so the nodes in the graphs represent "users" (defined as heuristically aggregated clusters of Bitcoin addresses). In such a graph the indegree of a node represent its number of incoming transactions. Informally it can be seen as the number of "payments" the user has received. As we have noted in the previous chapter, users with high degree often also have high values of other centrality measures and high economical weight in the ecosystem as well (i.e. they are among the richest users).



(a) (b)

Figure 3.1: Indegree frequency distribution of the users graph in December 2015 (a) and average distance of the users graph for 20 snapshots of the network, equally spaced from the beginning of 2013 to the end of 2015 (b).

The only spikes satisfying Definition 1 are the ones corresponding to indegree 708, 709, 771, and 772. The corresponding $I(x)$ values are between two and three orders of magnitude above the value expected. We have analysed the nodes of the network corresponding to these peculiar degrees.

To study the selected outliers, we started from a manual inspection of them. To do so we retrieved in the graph the clusters with indegree 708, 709, 771 and 772 and isolated their neighbourhood and transactions history. This provided us with 1647 clusters to analyse. We noticed that many of them had almost consecutive identifiers in our graph. According to our clustering algorithm, that happened because the oldest addresses contained in each of those clusters appeared for the first time in the blockchain together as destination of a payment in a unique transaction.

Looking for the first appearance of a sample of those addresses in the blockchain we noticed some peculiar transactions. One of these, for instance (Transaction hash `35dead89c059e846e2013a06a70cd84a7ba0f80da7741c283d6efd573e0a7319`) has one input and 101 outputs paying 0.00001 BTC to each one of the outputs except one, which is filled with the change (minus the fees). The address containing the change is then used to perform an analogous transaction leaving the change in a new address and so on. Basically the behavior of the transaction creator is to create a chain of transactions, where a transaction at each step pays a constant amount of 0.00001 BTC to some addresses (mostly chosen among the addresses cited in the previous paragraph) and leaves the change in an intermediary address used as input for the next hop in the chain. A chain ends either when the funds in the last change address are used for a transaction without this particular structure or when the input funds are completely spent and no change address is created in the last transaction of the chain. We also noted that the output addresses receiving 0.00001 BTC were addresses for the most part identifiable with users from the `bitcointalk` forum (indeed, the forum users had specified those addresses in their signatures), reused in each transaction of a single chain.

We will use these empirical considerations to model interesting transactions in Section 3.3.

3.2.2 Diameter

We noted in the previous chapter that the average distance in the giant connected component of the undirected users graph remains small over time (as shown in Figure 3.1(b), which reports the average distance of 20 snapshots of the networks equally spaced from the beginning of 2013 to the end of 2015), despite the increase in the number of nodes and arcs. That is the expected behaviour in a small world network as noted in other real world networks. In such networks we expect the diameter to be of logarithmic size in the size of the network [90], but in our graph the diameter is quite long, i.e. 2050 and constant over time. If compared to other real world undirected networks, our diameter value is much higher than expected [166, 49]. This fact contradicts the small-world hypothesis and the scale-freeness of the network, for which a short diameter is in general one of the well-established requirements [90].

It is worth observing that the very high constant diameter compared to the low average distance is a good hint of its artificiality. Since the diameter is constant it means that its path is very old, because it already existed during the first time snapshot (i.e. before January 2013), and no longer path is subsequently created. This suggests that the high length of this path was an exception rather than an usual property of the paths of the graph. In order to support this, we have approximated the distribution of distances using HyperANF [44] (using relative standard deviation: 13.18%)[3], obtaining that just 0.78% of the pairs of nodes are at distance greater than 38.

We remark that a path in the users graph represents a chain of value exchanges between users. If we consider a directed path, it represents the flow of value from source to destination through intermediate users. So the diameter of the undirected graph represents the longest economic interaction between a couple of users in the graph. Intuitively having a very long shortest path between a couple of users means that they have a lot of intermediaries between them in their exchanges. We can informally say that the closeness between two users in the graph represents how well connected they are in their local sub-economy.

To better understand the causes of the high diameter, we manually analysed the diameter path in our undirected users graph. The graphical representation of the corresponding directed path is depicted in Figure 3.2.

As we can see in Figure 3.2, the diameter path is mostly made of transactions with a very recognizable pattern. Those patterns are only sketched in Figure 3.2 and are zoomed in Figure 3.3, where N represents the total number of transactions. Such transactions have only one input with an arbitrary amount, paying exactly one satoshi (10^{-8} BTC which is the smallest amount of value a Bitcoin is divisible

[3] An exact computation is prohibitive as it requires $\omega(n^2)$, where n is the number of nodes.

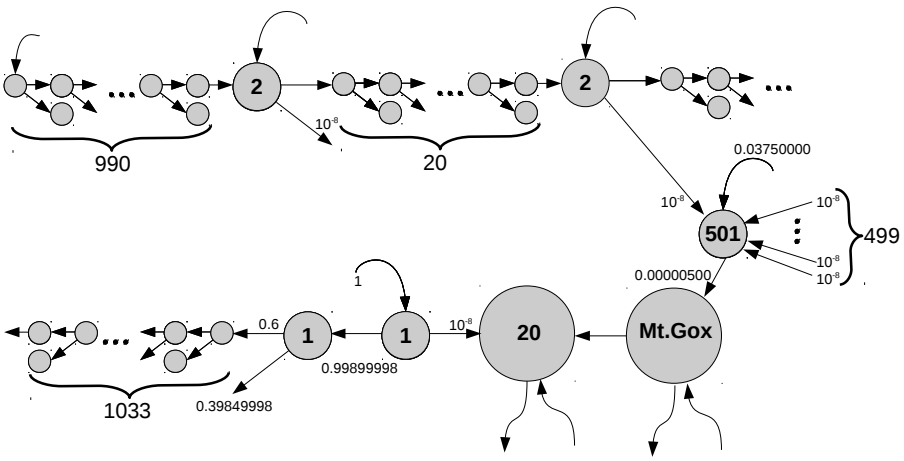


Figure 3.2: Scheme of the diameter path. Each cluster is labeled with the number of addresses it contains, or with a name if it is a well known cluster. Each arc might be labeled with the amount it is transferring, expressed in BTC.

in) to one address and the left over value (minus the transaction fee) to a newly generated address. This address is then used as input in a transaction with the same exact structure. The diameter starts with a chain of 990 of such transactions, paying in the end to a connection node. This connection node receives also value from another transaction and starts a new chain of 20 transactions of the same kind as above. The connection node is necessary because the value transferred from the last step of the chain is 0.00048001 BTC, that is lower than the fee of 0.0005 BTC usually paid by the transaction creator for a new transaction. This means that new value needs to be added from the outside to allow for the sum to keep moving in other transactions. The main purpose of a connection node seems to be to inject new liquidity after a chain has exhausted all its value (compared to the fee value chosen). The new shorter chain ends in another connection node that, in turn, pays exactly one satoshi to another cluster. This last cluster receives 499 other single satoshi payments and a bigger payment (0.03750000 BTC) before paying out all of the satoshi collected to Mt.Gox (0.00000500 BTC) and spending the rest of the value as fee. This intermediary node operates as a funnel, collecting the dust generated by the chains. We point out that Mt.Gox was the biggest exchange at the time of that transaction (18 October 2012), and, as such, the biggest hub in the network [181]. Mt.Gox then makes some big payments to a cluster that takes part in many high value transactions (including one of 900 BTC). This cluster is also the recipient of a single satoshi payment from a connection node that pays also a cluster that starts a new chain of 1 033 transactions of the same kind as the previous two.

We can notice a resemblance between these chains observed in the diameter and the ones observed in Section 3.2.1. In the next section we will give formal definitions to model both these observed transaction schemes (in Section 3.3.4 and Section 3.3.1

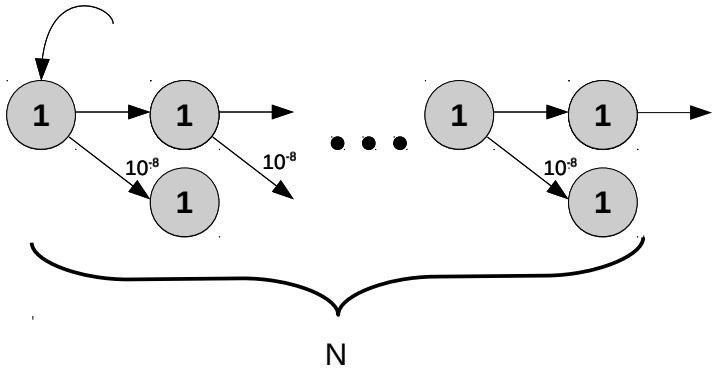


Figure 3.3: Scheme of the interesting chain pattern found in the diameter path. Each cluster is labeled with the number of addresses it contains. Each arc might be labeled with the amount it is transferring, expressed in BTC. The number under the curly bracket specifies how many transactions are part of the chain.

respectively). We will investigate their meaning and try to explain their purpose in Section 3.4.

3.3 Interesting Transaction Schemes

In this section we will model interesting transactions and relative structures. We will show in Section 3.5 how this transaction structures are responsible for the anomalies observed in the previous section.

Let us now introduce the concept of *suspicious transaction*.

Definition 2 *Let A be the set of all addresses present in the blockchain. Given $\alpha \in \mathbb{R}, k \in \mathbb{N}$, and a transaction t modeled as a tuple $(In, Out, InAmount, Fees)$, where:*

- $In \subseteq A$*;*
- *Out is a multiset of couples (o, b) where $o \in A$ and $b \in \mathbb{R}$, where b corresponds to the amount paid to address o by In;*
- $InAmount \in \mathbb{R}$*;*
- $Fees \in \mathbb{R}$*;*

*we say that t is an (α, k)-*suspicious transaction *if it satisfies the following properties:*

- $|In| = 1$ *;*
- $|Out| \geq k$ *;*

- $|\{(o, b) \in Out \ : \ b \neq \alpha\}| \leq 1$.

In other words, a (α, k)-suspicious transaction is such that the only input address pays all the output addresses, which are at least k, α BTC except at most one, which can be the recipient of an arbitrary amount. We call this particular address *change address*, i.e. a is a change address if a is s.t. $(a, b) \in Out$ and $b \neq \alpha$. We call *common output* any output containing an address that is not a change address, and α *common amount*. By definition, in each (α, k)-suspicious transaction can exist at most one change address. We write $(*, k)$-suspicious transactions to refer the union of (α, k)-suspicious transactions for any $\alpha \in \mathbb{R}$.

In the rest of this sections, we will define some interesting transaction structures made of (α, k)-suspicious transactions for specific values of α and k. We will then show in Section 3.5 how these structures are linked with the observed anomalies.

3.3.1 bps-transactions

The manual observations of the indegree frequency distribution outliers in Section 3.2.1 led us to find an interesting transaction pattern. The observed transactions had all a single input and multiple outputs paying all a common amount of 0.00001 BTC but at most one receiving the change instead. We labelled transactions with this behaviour *base-pseudo-spam* (BPS-transactions) and they can be modelled as suspicious transactions as follows:

Definition 3 *A* BPS-*transaction is a* $(0.00001, 2)$-*suspicious transaction.*

We will explain in the following Section 3.4 why we chose the term "pseudo-spam" to define such transactions.

As we noticed in Section 3.2.1, the interesting behaviour is not only about the transactions themselves, but rather about their use as links in a chain. For this reason, we define a "base-pseudo-spam chain" (BPS-chain) as follows.

Definition 4 *Given two* BPS-*transactions* t_i *and* t_j *we say that* t_i *is* joint *to* t_j *if and only if the unique input address of* t_j *is the change address of* t_i *and the input amount of* t_j *is the value paid to the change address in* t_i.

Definition 5 *A set* C *of* n BPS-*transactions (i.e.* $|C| = n$*) is called a* base-pseudo-spam chain *(*BPS-*chain) if and only if there exists an ordering of elements of* C *i.e.* $C = \{t_1, t_2, ..., t_n\}$ *so that* $\forall i$ *s.t.* $1 \leq i \leq n - 1$ t_i *is joint to* t_{i+1}.

Given a set T of BPS-transactions, it can be partitioned in BPS-chains following the previous definition. Of course we are interested in the minimal partition of T, i.e. the one dividing the set by building the longest possible chains. So, in the rest of this chapter, we only consider the BPS-chains obtained from the smallest partition

of T. In other words whenever two BPS-chains can be merged we merge them[4]. In the rest of this chapter we will call "singleton" a chain of length one (i.e. a chain made of a single transaction).

3.3.2 ps-transactions

In the previous sections we have defined what is a BPS-transaction and a BPS-chain. The definition of a BPS-transaction was given keeping into account our practical consideration that the manually inspected transactions in Section 3.2.1 pay an amount equal to 0.00001 BTC to each output. We want now to model a more general class of chains based on the experimentally observed structure but not limited to the experimental value found in the case study. In other words the interesting information obtained from the manual inspection is not the value of the amount paid as output but rather the transactions structure. Taking into account this observation, we consider different transactions sharing a similar structure to the BPS-transactions but having an arbitrary amount spent by the common outputs (called "pseudo-spam" transactions).

Definition 6 *A* PS-transaction *is a* $(*, 3)$*-suspicious transaction.*

It is worth observing that not all the BPS-transactions are PS-transactions. Even if the concept of PS-transactions is meant as a generalization of BPS-transactions, formally PS-transactions are not a generalization of BPS-transactions due to the different minimum number of outputs. Indeed, we point out that we need to consider transactions of at least three outputs to be able to distinguish between the common outputs and an eventual change address. This happens because if the common amount value is arbitrary we can no longer distinguish a common amount output from a change output in a transaction with two outputs only. We also further restrict ourselves to only considering PS-transactions with common output value smaller than 1 BTC.

We define a PS-chain similarly to Definition 5 but using PS-transactions instead, and requiring the common amount to be the same for all transactions part of a chain:

Definition 7 *A* Pseudo-Spam chain *(*PS*-chain) is a sequence of at least two* PS*-transactions,* $t_1, \ldots, t_h$*, with* $h > 1$*, so that:*

- *the unique input address of the* i*-th transaction is the change address of the* $(i-1)$*-th transaction;*
- *there exists* $c \in \mathbb{R}$*, so that for any* PS*-transaction* t_i *in the sequence, all output amounts in* t_i *except the change address output are equal to* c *(common amount).*

[4]Following the two previous definitions two ordered BPS-chains $C_1 = \{a_1, ..., a_n\}$ and $C_2 = \{b_1, ..., b_m\}$ can be merged if and only if a_n is joint to b_1 or b_m is joint to a_1.

3.3.3 Impact Measure

Given any transaction chain we want to define a measure of how "impactful" the chain is for the graph. By "impact" of a chain we informally mean the weight of the chain measured as the number of nodes it reaches weighted by the number of times it reaches them. So we consider more "impactful" or "disruptive" a chain that touches a lot of nodes, each multiple times. This is meant to reflect the "spam" nature that we aim to model (see beginning of Section 3.4). Formally we define the *impact measure* of a chain as its average number of common amount outputs times its length, or in other words the total number of outputs (excluding change addresses used as intermediary chain links) of all the transactions included in the chain. Since we are only interested in impactful transaction chains, to help prune false positives we can chose a threshold for the chain impact measure below which the chain are to be considered uninteresting and we can prune the PS-chains set accordingly. This leads us to define the PS-set notion:

Definition 8 *Given a* PS*-chains set C and a threshold $r \in \mathbb{R}$ we define as* pseudo-spam set *(*PS*-set) the set of transactions t_i such that there exists j with $t_i \in c_j$, $c_j \in C$, $|c_j| > 1$, and impact$(c_j) \geq r$, where impact(c_j) is defined as the sum of the number of common outputs of each transaction $t_u \in c_j$.*

In other words, given a threshold, a PS-set derived from a PS-chains set is the set of all the PS-transactions belonging to a chain in the set that is not a singleton and has a chain impact measure greater than the threshold.

3.3.4 gps-transaction

During the manual inspection of the diameter path in Section 3.2.2 we have found out the existence of long chains of homogeneous transactions (depicted in Figure 3.3) that are responsible for the length of the diameter path. The goal of this section is to model such chains and transactions as suspicious transactions. To do so we introduce the concept of *generic pseudo-spam transaction.*

Definition 9 *A* generic pseudo-spam transaction*, (*GPS*-transaction) is a* $(*, 2)$*-suspicious transaction.*

First we note as the transaction pattern we want to model is similar to the one modelled by the PS-transactions definition, but with a major difference in the minimum number of outputs required. By relaxing the definition to account for transactions with only two outputs we accept the fact that we cannot know, given a single transaction with only two outputs, which output represents the change address. Moreover the above definition is satisfied by any transaction with exactly one input and two outputs, since the two output amounts are either equal or different, and the definition is satisfied in both cases.

We also note that we called the new transaction pattern *generic* PS-*transaction* because it is a generalization of both the two previous PS-transaction and BPS-transaction definitions. If a transaction satisfies Definition 3 or Definition 6 then it also satisfies Definition 9. In other words both the sets of BPS-transactions and PS-transactions are subsets of the set of GPS-transactions. This could make us question the need for the previous PS-transaction definition, but we should not forget that the two kinds of transactions derive from two very different sets of observations in Section 3.2. Despite their definitions being so close, we will show how the two kinds of transactions can have different economical meanings in Section 3.4. Furthermore we will prove that PS-transactions and GPS-transactions are responsible for indegree outliers and the high diameter respectively in Section 3.5.

We define a *generic pseudo-spam chain* the same as we defined PS-chains in Definition 7.

Definition 10 *A* generic pseudo-spam chain *(*GPS-*chain) is a sequence of at least two* GPS-*transactions,* $t_1, \ldots, t_h$*, with* $h > 1$*, so that:*

- *the unique input address of the* i*-th transaction is the change address of the* $(i-1)$*-th transaction;*
- *there exists* $c \in \mathbb{R}$*, so that for any* GPS-*transaction* t_i *in the sequence, all output amounts in* t_i *except the change address output are equal to* c *(common amount).*

We want to remark the importance of the topological information given by chaining transactions together. Definition 9 is too general to provide useful information by itself. In fact, 50 432 329 transactions over the 99 602 440 transactions in our dataset (i.e. 50.6%) satisfy Definition 9 (and so are labelled as GPS-transactions). This is mostly due to the fact that transactions with exactly one input and two outputs are really common. 49 284 766 transactions in the dataset have just one input and exactly two outputs with different amount. Despite that, only 5 611 330 transactions (i.e. 5.6% of all transactions) are part of a chain of a GPS-chain. So even if these transactions are very common in the dataset only few of them compose in chains which are interesting by our standards.

3.4 On the Economical Meaning of Our Transaction Schemes

First we want to clarify the reason behind the naming adopted for the transaction schemes introduced in Section 3.3 (derived from the observations in Section 3.2). We decided to use the term "pseudo-spam" to label those transaction models that seemed to have a "spam" effect on the graph measures, i.e. transactions that, even if minoritarian in the dataset, had a macroscopically visible effect on the graph

properties, because of how they were formed or because of how they were combined together.

We should now remark that in the Bitcoin protocol transactions are the only mean available to update the state of the system. This means that transactions do not necessarily represent "payments" in the traditional sense, they can be used as funds management tools as well. In particular long chains of transactions are often employed as payout methods from many different entities in the Bitcoin ecosystem, such as mining pools, exchanges or mixing services. For example, it would not be surprising for a fixed payout amount mixer service to use a GPS-chain structure for payouts [191]. Another example of a chain of transactions with a GPS-chain structure can be a fixed amount peeling chain [230], a known obfuscation technique aimed at trying to weaken deanonymization attempts. This does not contrast with what we mean by "pseudo spam" transactions, the term "pseudo spam" is not linked in any way to a malicious or uncommon behaviour.

Some of the transactions considered in this chapter have not passed unnoticed. For example, the transactions manually observed in Section 3.2.1 target what, at first glance, seems like a random selection of addresses in the blockchain. Bitcoin users in the past have noticed receiving unexpected payments from such transactions and some interest has sparked around them. Unfortunately, there has not been a clear accepted explanation of the goal of such transactions yet. In the following we explore some possible existing conjectures, showing that none of them is able to fully explain the purpose of transactions modelled by our suspicious transaction definition.

A first conjecture may be that these transactions are part of an attack on users pseudonymity, as an attempt to link addresses ownership. In fact the amounts sent are so tiny that in order to be spent they must be first combined with other funds in a multi-input transaction. This would potentially reveal new linking for the multi-input clustering heuristic (see Section 3.1) increasing its effectiveness. Even if this theory sounds reasonable, it does not seem to be applicable to our observed real use cases. In fact not only the targets of our manually observed suspicious transactions in Section 3.2.1 are not picked at random from the blockchain but derived from a very close set of users belonging to the `bitcontalk` forum, but, more importantly, the transactions pay the same amount to any address multiple times. For an attacker it would make little sense to send funds (hence spending them) to the same address a lot of times and would be more efficient to send those funds to different addresses instead, because this would increase its probability of triggering a funds consolidation while minimizing the cost of the attack.

Another possible conjecture is that those transactions are used as part of a spam attack, to fill the blockchain space with useless data. But this is arguably not true since most of those transactions pay a regular fair fee to be included in the blockchain and so they have the same right to be included as any other transaction. Note that we perform a transaction analysis based on the blockchain information, so we only consider the permanent effect of transactions. In the case of BPS-transactions, the

kind of transaction observed in Section 3.2.1 can be effectively used to perform a live spam attack to rapidly fill the users pending transactions lists, as historically really happened during the flooding attack of July 2015 [27]. This is supported by the fact that most BPS-transactions were performed at that time. But live spam attacks by themselves are expected to leave little to no sign on the blockchain. Interestingly enough the attack happened during the same time as the hardly debated discussion about scalability issues of Bitcoin block size, so it might have been an attempt of an unknown party to practically prove the inadequacy of the fixed block size. This same attack has been independently observed and studied in [179], reaching the same conclusion.

Another possible interpretation would be that these transactions are used for advertising. By using vanity addresses or inserting human readable messages in the transactions it is possible to use a transaction to cheaply save an advertisement message in the blockchain forever. By including in such transactions the largest possible number of outputs one may attempt to increase the message visibility.

A famous example of these transactions arose to popularity during the Sochi Olympics, because two addresses (`1SochiWwFFySPjQoi2biVftXn8NRPCSQC` and `1Enjoy1C4bYBr3tN4sMKxvvJDqG8NkdR4Z`) started sending thousands of transactions paying exactly 1 satoshi (0.00000001 BTC) to what seemed like random addresses read directly from the blockchain. Those transactions paid no fees and so only few of them were actually saved in the blockchain but they remained for hours in the users wallets as unconfirmed transactions, gaining a lot of visibility [226, 31]. It seems difficult to think that this was part of a deanonymization attack since most of the transactions never became part of the blockchain and so could not be spent to possibly reveal addresses linking. It might have been considered a spam attack but only limited to the live network (by filling the unconfirmed transaction lists of users with useless data) but it left very little effect on the blockchain since few transactions were actually included. So the most plausible theory seems to be that it was part of a temporary spam advertising campaign, and a successful one since most Bitcoin users received the message to "Enjoy Sochi" with very little cost fro the authors. The cost was so little since very few transactions were accepted in a block (hence actually spending the used funds) and the 0.00000001 BTC payments carried so little value to do not matter anyway.

Whatever is the reason for this kind of transactions, we argue that they should be considered artificial transactions (i.e. not representing a value exchange between users). We have observed that in practice outputs of suspicious transactions are often spent (i.e. used as inputs in the next step of the chain) inside the same block. To not wait for confirmations before spending an output is equivalent to blindly trusting the corresponding paying entity to be honest. If this blind trust is shared by a lot of addresses in the same chain, it is a good indication of the presence of a single entity (that, of course, needs no trust in itself) controlling such addresses and so the resulting chain. Furthermore in general we noted that GPS-chains are created much faster than average. Note that, due to the random nature of new

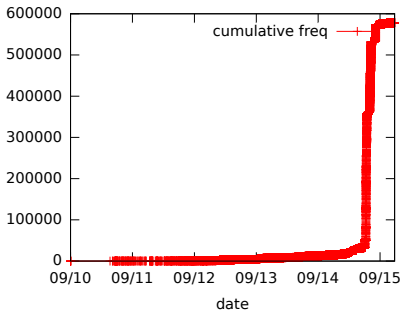


Figure 3.4: Timestamps Cumulative Frequency Distribution of BPS-transactions.

blocks creation, it is sometimes preferred to measure the time in the blockchain as block height difference between blocks. The average time that a transaction output wait before being spent is 1 326 516 seconds (or 2 413 blocks, if measured by block height difference) considering all spent outputs in the dataset. But if we compute the same mean for change address outputs part of GPS-chains only we obtain 47 736 seconds (or 88 blocks). This shows how GPS-chains are created much faster than general transaction trees. Indeed the topology of the chains we have observed seems to suggests that the transaction purpose is to obtain some kind of side real world effect rather than to transfer value between addresses. This is clearly obvious for the Sochi example since the fair fees cost of the transaction would exceed the value effectively transferred. The same can be said for our manual inspected transactions in Section 3.2.1. For example, in one of those the fee was 0.0007184 BTC, hence seventy times the single amounts transferred and 41.8% of the total value actually spent by the transaction.

The same reasoning can be made about the GPS-chains found in the diameter in Section 3.2.2. Each GPS-transaction in those chains pays 0.00000001 BTC out while spending a 0.0005 BTC fee. So the value spent as fee is 50 000 times larger than the value paid out. We also point out the peculiar common amount of those GPS-chains. Usually the aim of one satoshi (i.e. 0.00000001 BTC) transactions is not to transfer value. One well-known example comes from blockchain betting services (such as SaoshiDICE [30]) that rose to popularity during the same year of the creation of the observed diameter chains (2012). In such services the user sends some value to a publicly advertised service address, and the service pays back the user with the winnings or sends him back one satoshi to notify he has lost. Aim of the one satoshi transaction is only to inform the user of the loss rather than giving him a consolation price, since the value paid is in practice useless.

3.5 Relating Transaction Schemes to Anomalies: Experimental Evaluation

In this section we will experimentally show how the transaction structures defined in Section 3.3 can be used to explain the anomalies observed in Section 3.2.

3.5.1 bps-chains and the indegree frequency distribution

In Section 3.3.1 we have defined BPS-transactions to model the transactions manually observed in Section 3.2.1. Applying Definition 3 to our dataset we labeled 578 316 transactions as BPS-transaction, out of the 99 602 440 multi-input multi-output transactions contained in our database. The transactions vary a lot (considering most of the transactions features as the number of outputs or the fees paid), but an interesting behavior can be seen analysing the timestamps cumulative frequency distribution among those transactions. As shown in Figure 3.4 we can see a steep step during July 2015 showing that most of those transactions were performed at that time. This is consistent both with our observations, since the transactions of our case study took place during July 2015, as well as with the existence of an historically recorded flooding attack happened during the same period [27].

Considering as T the set of all the BPS-transactions, we merged the BPS-transactions in chains following Definition 5 obtaining 24 381 BPS-chains. To prune the BPS-chains multiset from false positives we eliminated from the multiset all the singletons, hence discarding all the BPS-transaction that were not part of any chain. This left us with 3 805 BPS-chains. In the following we report some basic statistics of the BPS-chains we found.

Average Number of Outputs. We observed the cumulative frequency distribution of the average number of outputs (excluding the change address linking to the next step in the chain) in each chain, which is shown in Figure 3.5(a1). We can notice how a large number of chains (i.e. 2 240) has exactly one single output address (excluding the change address). If we consider the cumulative frequency distribution ignoring this special case, hence ignoring single common amount output chains, we can notice a steep increase around 50 and a less marked increase around 100. So the value 50 seems to be the preferred average number of outputs for BPS-chains. The transactions we have manually examined in Section 3.2.1 had 100 outputs excluding the change address, corresponding to the second most common value (excluding one). This means that the observed transactions even if they are not the most representative sample are still not an isolated case among BPS-chains regarding the average number of outputs. More precisely approximately 7% (if we don't count the single output chains) of the BPS-chains found share this behavior.

Chain Lengths. If we consider the cumulative frequency distribution of the lengths of the BPS-chains found, shown in Figure 3.5(b1), we can notice very high initial values as well. More precisely, the chains of length two are 39.3%, while the

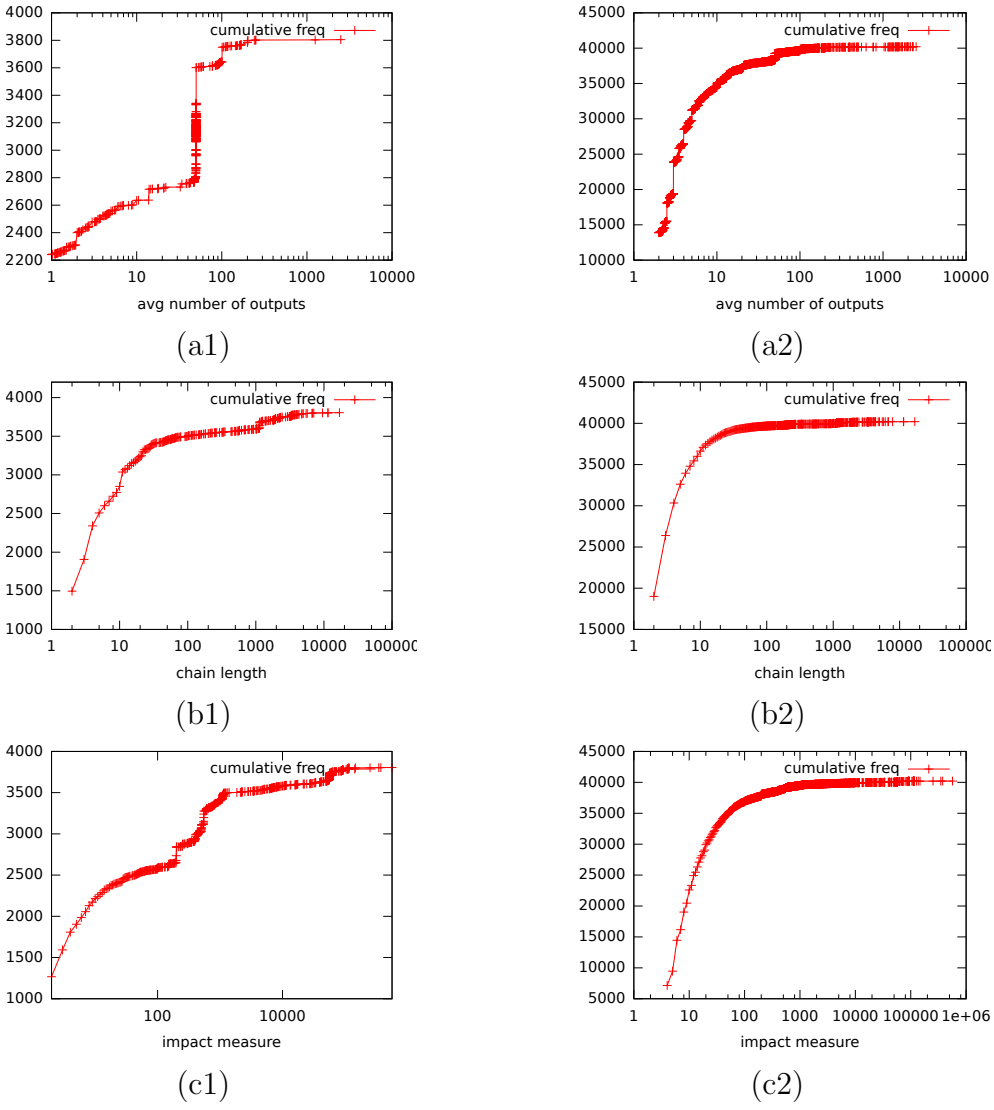


Figure 3.5: BPS-chains statistics (a1,b1,c1) and PS-chains statistics (a2,b2,c2)

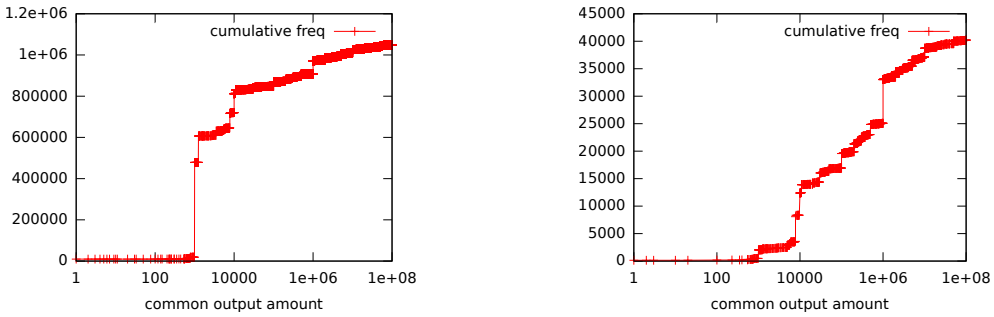
chains of length at most three are already more than 50%.

Chain Impact. The small average number of outputs and the short length of many chains suggests that we found a lot of chains with a very low overall number of outputs. The the impact measure measures the overall number of common outputs (see Section 3.3.3) and its cumulative frequency distribution is shown in Figure 3.5(c1). The higher this value is, the more "disruptive" the chain can be considered for the graph. From the plot we can observe as 33.3% of all the chains share the minimum value, it means that one third of all the chains has length two and only one output is not a change output in each of its two transactions. We think that chains with so little values may result from a lot of "normal" use cases rather than artificially constructed. Hence, for small values of the impact measure we should not consider the chains interesting. Even if those chains are not naturally occurring but deliberately created, their impact on the network is limited (due to the minimal number of outputs involved) and not statistically relevant (since they represent 0.026% of all the multi-input multi-output transactions).

3.5.2 ps-chains and the indegree frequency distribution

In Section 3.3.2 we have defined PS-chains (and PS-transactions) to model the general structure of BPS-chains independently of the common amount used. By applying our classification to the blockchain we found 1 050 783 PS-transactions that could be joined in 149 328 PS-chains. Among these, 40 208 PS-chains were not singletons.

If we perform the same analyses on some basic statistics of the PS-chains found as we did for the BPS-chains in the previous section, we obtain similar results. The plot of the cumulative frequency distribution of chain lengths shown in Figure 3.5(b2) and number of outputs (excluding change addresses) shown in Figure 3.5(a2) show a similar behaviour as in Section 3.5.1, with 47.3% of the chains having length two and 34.6% of the chains having the minimum number of outputs. This suggests that our case study was a good approximation of the general "pseudo-spam" phenomenon modelled by PS-chains. If we evaluate the impact measure cumulative frequency distribution for PS-chains we obtain a similar but smoother plot, shown in Figure 3.5(c2). For PS-transactions and PS-chains we can also consider a new parameter that is the common amount value of transactions and chains. The common amount value cumulative frequency distribution for PS-transactions found is depicted in Figure 3.6(a). We can immediately observe how the common amount value used in our case study (0.00001 BTC) in Section 3.2.1 is the most frequent value for PS-transactions, covering 43.8% of all such transactions. We also note that all of the highest frequency common values are all "clean" values (e.g. 1, 1000, 1250, 2750, 3000, 3500, 10000). This is compatible with human designed transactions rather than random purchase transactions, since prices are usually expressed in traditional fiat currencies such as USD or EUR, and their change in BTC is rarely a "clean" number. In Figure 3.6(b), we show the common amount values cumulative frequency distribution of the PS-chains found. In this graph the highest frequency



(a) (b)

Figure 3.6: Common output amount (expressed in 10^{-8} BTC) cumulative frequency distributions for PS-transactions (a) and PS-chains (b)

values are clean numbers (e.g. 1000, 10000, 100000, 200000, 500000, 1000000) as in the previous graph but the value 1000 (corresponding to 0.00001 BTC) has a smaller importance. This means that a lot of the PS-transactions with this common output value were joined in single long chains.

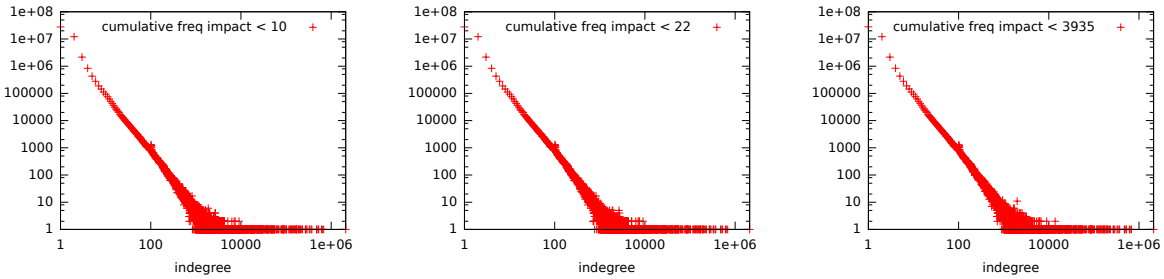
3.5.3 Indegree frequency distribution anomalies explanation

As a result of the observations in the previous sections, in this section we aim to prove the following property.

Property 3 *The outliers observed for the indegree frequency distribution are caused by few* PS*-chains.*

We will show that these few PS-chains target a small set of addresses increasing their corresponding cluster's indegree, explaining the origin of the four outliers observed in Section 3.2.1. It is worth observing that not all of the output addresses of PS-chains are among those four outliers. Those other addresses do not stand out because they are part of already popular clusters, and so their indegree is marginally affected by those transactions while the pseudo-spam effect is more visible in other unpopular addresses. We also observe that we shall not expect all of the clusters with an outlier indegree value of 708, 709, 771 or 772 to be artificially inflated. In fact, following the power law, it is natural to expect the existence of a little number of clusters with these indegrees.

We start by checking if the clusters marked as outliers have at least one address that appears as output in a PS-chain. We find out that 1 630 over 1 647 clusters satisfy this. It means that only 17 clusters are not affected by the PS-chains. These findings are consistent with what we expected. More precisely, if we restrict just to cluster not involved in PS-chains we obtain an *outlier-free* indegree frequency distribution. This alone is of course not enough to prove our supposition yet. We have only observed that all the outliers take part in a PS-chain but we still have to



(a) (b) (c)

Figure 3.7: Comparing the indegree frequency distribution of the users graph pruned of the transactions belonging to the PS-set for threshold values of 10 (a), 22 (b) and 3935 (c).

prove that the PS-chains are the sole cause of those outliers. To do so we employ the notion of PS-set defined in Section 3.3.3. To check if a PS-set alone is causing the indegree frequency distribution outliers we re-compute the indegree frequency distribution of the users graph, ignoring the transactions belonging to the PS-set obtained from the PS-chains candidate set obtained in the previous Section 3.5.2, for increasing values of the threshold.

To choose the threshold values we look at the plot of the chain impact measure, shown in Figure 3.5(c2), and we observe that more than 50% of the chains have an impact value smaller than 10, more than 75% have an impact value smaller than 22, more than 90% have an impact value smaller than 73 and more than 99% have an impact value smaller than 3935. So we choose those three values (10, 22 and 3935) as thresholds to obtain a PS-set. This results in the indegree frequency distributions depicted in Figure 3.7. As we can see the outliers disappear for all the values of the threshold considered without macroscopically affecting otherwise the overall frequency distribution (see Figure 3.1(a) for a comparison). This not only proves Property 3, but also means that the outlier generating chains of our case study are among the chains with largest impact, and so among the longest and/or with most outputs chains[5]. This explains why those chains are the ones that so macroscopically affect the indegree frequency distribution of the entire network, enough to cause outliers in said frequency distribution. Note that even if only the highest impact chains macroscopically affect the indegree frequency distribution all the PS-transactions in the PS-set do influence it. So including also lower impact chains helps cleaning the indegree frequency distribution from artificially skewed values. Of course the lower the impact value used as threshold the more probable is the presence of false positives in the set. A trade-off between the two has to be found.

[5]Because a PS-set with threshold t is always subset of any PS-set derived from the same PS-chains set and threshold t' such that $t' < t$.

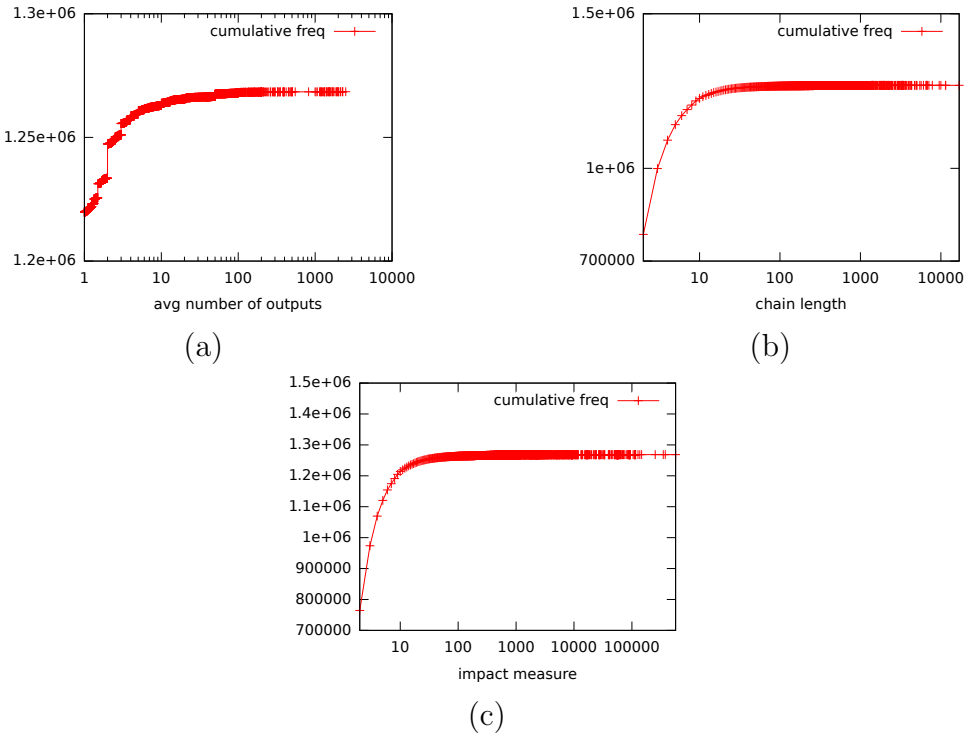


Figure 3.8: GPS-chains statistics

3.5.4 gps-chains and the diameter

We have given in Section 3.3.4 the definition of GPS-chains to model the long chains of transactions found during the manual inspection of the diameter path in Section 3.2.2 (see Figure 3.3). In our dataset there are 1 268 473 GPS-chains satisfying Definition 10. For a comparison with the PS-chains found in the first part of this chapter we evaluated the cumulative frequency distribution of the same measures observed in Section 3.5.2, obtaining the results shown in Figure 3.8. If we compare Figure 3.8(a,b) with Figure 3.5(a2,b2) we can see how the chain length and average number of outputs (at each step of a chain, excluding any change address) cumulative frequency distributions show a similar trend. We can also note that we have an high percentage of chains showing the minimum values for such measures as in the PS-chains case. 62.0% of the GPS-chains found have length 2, moreover 97.1% have length smaller or equal than 10. Surprisingly 96.2% of the chains have exactly one output at each step of the chain. This means that a lot of GPS-chains follow the same behavior of the chains we noticed in the diameter in Section 3.2.2. Moreover, since PS-chains are also GPS-chains, and PS-chains can not have only one output at each step (by definition the minimum is two), we can conclude that most of the new chains found show this different behavior (regarding the number of common amount outputs) compared to the PS-chains. Of course we can also say that the GPS-chains are mostly not also PS-chains, but that was already obvious by noting

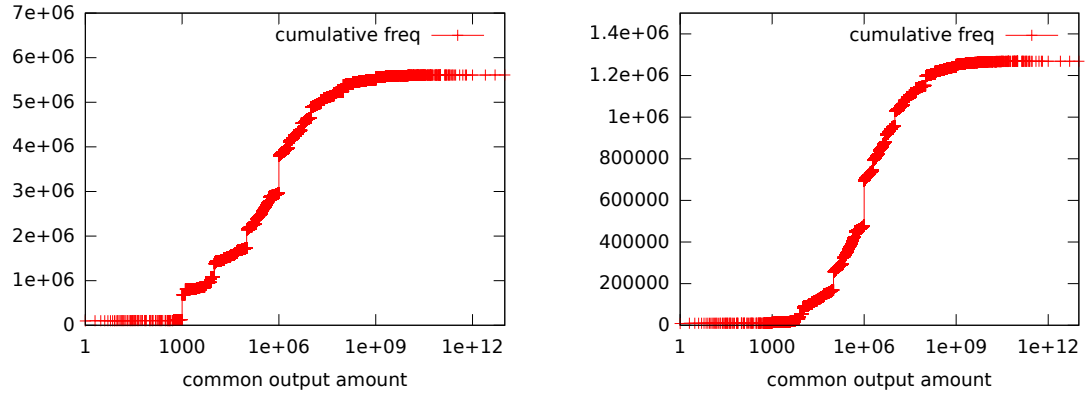


Figure 3.9: Common amount (expressed in 10^{-8} BTC) cumulative frequency distributions for GPS-transactions (a) and GPS-chains (b)

that we have found a number of GPS-chains thirty times higher than PS-chains. The impact measure cumulative frequency distribution (Figure 3.8(c)) shows a similar trend to Figure 3.5(c2) but with higher initial values as we expected considering the higher initial values of both average number of outputs and chain length.

In Figure 3.9 we show the cumulative frequency distributions of the common amount values of GPS-transactions and GPS-chains found. We can see a similar trend to Figure 3.6, even if the frequency distributions in Figure 3.9 are smoother. We can still notice how the majority of transactions (53.8%) uses a common amount of 0.00001 BTC similar to what happened for PS-transactions. We also point out how we can easily detect clear steps in Figure 3.9(b), consequence of human designed transaction amounts, as noted in Section 3.5.2.

3.5.5 Diameter anomaly explanation

In this section we aim to prove the following property.

Property 4 *The high diameter is heavily affected by few* GPS*-chains.*

We have shown the diameter path in Figure 3.2 and we have already noted how its high length is due to three interesting transaction patterns. Furthermore the three chains observed in the diameter path satisfy the GPS-chain Definition 10. Now we should remember that we are interested in studying the users graph, that is the graph derived from clustering together different addresses belonging to the same user. The meaning of a GPS-chain is to pay out some common amount to the output addresses and use a change address at each step to keep the value flowing to the next transaction of the chain. Under this assumption the change address can be seen as a temporary address, or, in other words, a kind of placeholder to allow for the chain to be built. Furthermore, it is reasonable to assume that each chain is

built by the same entity due to their peculiar homogeneity. This means that each change address in the chain is controlled by the same entity (because it is spent by it). These observations imply that we can consider all the change addresses in a chain as belonging to the same user that started the chain. This means that we can cluster all these addresses together in a single cluster (see Section 3.1 and Section 3.4). This has the same effect as *collapsing* a chain, i.e. to replace a chain with a single transaction paying at once all the common outputs in all the transactions at each step of the chain (hence ignoring the intermediary change addresses), as well as the last change address of the chain.

If we apply this new clustering step to our diameter path depicted in Figure 3.2 it becomes long only 9 instead of 2050, drastically reducing its length. To test if GPS-chains heavily affect the high diameter (hence proving Property 4) we applied this newly defined clustering step to the users graph, collapsing all the GPS-chains. We then computed the exact diameter of the new user graph by using the same algorithm ([49]) as the previous chapter. This new diameter has length 575, that is much shorter than the previous one, i.e. 2050.

The new diameter is still higher than we expected so we further investigated its structure. Our preliminary results show how this new diameter is artificially inflated as well by peculiar transaction patterns that can be grouped in chains different from GPS-chains. We plan to study those new interesting transaction patterns in our future work (see Chapter 4).

4

Conclusions and Future Work

Abstract

In this chapter we provide our concluding remarks about the work presented in this Part I. We show how the information obtained by the analysis performed in Chapter 2 and Chapter 3 can be used in conjunction with the deanonymization attack presented in Section 1.5 (i.e. combining blockchain analysis with heuristic rules clustering and real world identities information linking to obtain the Bitcoin identity graph). To this aim we present our ongoing research to strengthen the user graph analysis, to define new heuristic rules and to deploy an efficient Bitcoin network listener.

We have shown in Chapter 2 a scalable clustering algorithm able to support the construction of the Bitcoin users graph and a set of analyses, applied on the users graph produced by this algorithm, which allows to uncover several properties of the Bitcoin network. The users graph is derived from the transactions present in the blockchain until December 2015, this includes about 100 millions of multi input, multi output transactions. We have then presented a set of interesting properties of the Bitcoin network, like the "rich get richer" property and the existence of central nodes acting as hubs between different parts of the network. We have also verified that the clustering coefficient of the users graph is the same of order of magnitude of other complex networks [197], which highlights the complex nature of the Bitcoin network. During our analyses we have observed that the diameter of the Bitcoin users graph is much larger than the one of social networks, in spite of a small average distance between the nodes, and that the degree distributions have some outliers for some specific values. The study of the actual users behaviours leading to these peculiar properties has triggered the work presented in the subsequent chapter.

We started Chapter 3 by conjecturing that the interesting topological patterns causing the structural anomalies observed in Chapter 2 are due to unexpected users behaviours, not strictly related to normal economic interaction. By manually analysing the users graph we have found out that these phenomena are generated by peculiar chains of transactions. We have then formally characterized such chains and studied their impact on the dataset through automatic tools. We have also given possible interpretations of the purpose of the observed chains in the Bitcoin

ecosystem. At the end of the chapter we have shown how PS-chains and GPS-chains are not the only interesting repeating transaction patterns causing long chains in the graph. Deepening the research on this topic by further classifying other recurrent transaction patterns is an interesting line for future work. We plan to keep working on this topic to give more insights into the nature and possible semantic of PS-chains and GPS-chains, and to expand the analysis to include new types of interesting transaction patterns alongside their effect on the Bitcoin users graph. The preliminary results on this topic are shown in Section 4.1.

Another research task we are currently exploring is to strengthen our graph analysis. First we plan to achieve this by updating our dataset to the current Bitcoin blockchain, comparing the updated results to see how the measured graph properties have changed in the last two years. We also plan to widen the analysis by studying more graph properties. For example, in Chapter 2 we have computed some centrality measures to detect the most central nodes in the network. The results confirm the presence of a set of nodes, corresponding to very popular entities in the Bitcoin ecosystem, which are central according to several definitions of centrality. Moreover, we have seen that considering shortest paths or arbitrary paths to define centrality of nodes can significantly change the ranking. We conjecture that this is mainly due to the presence of many alternative paths and we expect to verify this conjecture, for instance computing the hyperbolicity of the network [47]. Finally, we also plan to extend our analysis to highlight the relation between the structure of the users graph and other interesting properties of the Bitcoin economy. For instance we are trying to understand how speculators can be detected by an analysis of the users graph, and studying the possibility of finding graph structures whose number of occurrences correlates with price trends.

As we have explained in the introduction to this Part I, the work shown in both Chapter 2 and Chapter 3 can be used to strengthen the Bitcoin deanonymization attack shown in Section 1.5.2. The deanonymization attack is based both on historical information stored in the blockchain and on the live information available in the Bitcoin network. In order to strengthen the blockchain side of the attack, i.e. the building of the user graph, it is necessary to define reliable heuristic rules. But heuristic rules are derived from the study of users behaviour on the network, and that is precisely what we have done in Chapter 3. In fact, detecting and isolating peculiar chains of transactions that are created by the same user allow us to define new heuristic rules to cluster together the chain intermediate steps. We are currently studying how effective can be heuristic rules based on the PS-chains and GPS-chains patterns found. An example of such rule proposal, named *chain heuristic*, is shown in Figure 4.1. This heuristic rule is based on GPS-chains and states that all the intermediate change address of a GPS-chain are controlled by the same entity creating (i.e. starting) the chain. This is consistent with our assumptions in Chapter 2 that the chain is created and continued by a single entity. In fact, if this is the case, the intermediate change addresses at each step are only expression of the need for a change address in Bitcoin protocol transactions, they do not represent autonomous

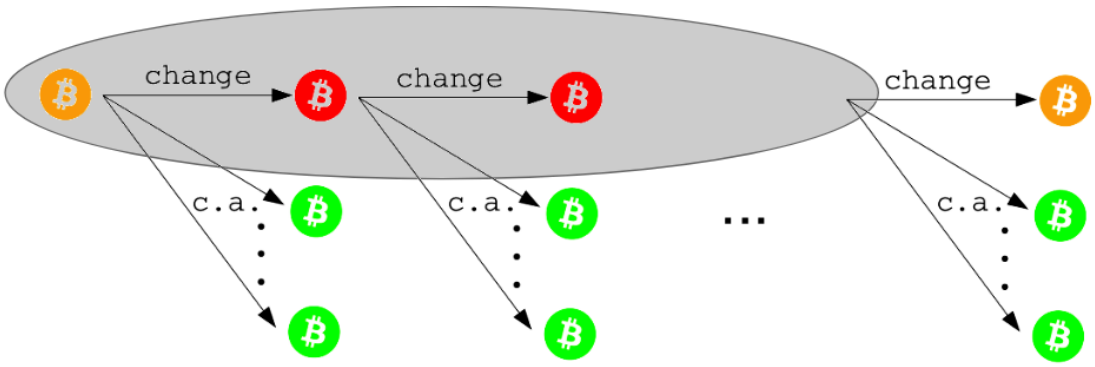


Figure 4.1: Scheme of the proposed chain heuristic rule. All the intermediate steps of a GPS-chain are clustered together with the address starting the chain.

entities receiving and making payments.

The proposed heuristic can be seen as refinement of the change address heuristic (see Section 1.5.2), since it basically exploits the information that a transaction belongs to a GPS-chain to clearly find out which is the correct change address among the outputs. This and other kinds of possible *chain heuristics* can, in fact, can be seen as multi hops change heuristic, since they use future information of the blockchain to find chains and retroactively cluster together addresses. The main issue in developing any new heuristic rule is that no ground truth is available to test the false positives and negatives performances of the proposal. We are currently studying different solutions to overcome this limitation and give reliable performance evaluations about our proposed novel heuristic rules.

The last topic we are also currently working on regarding Bitcoin anonymity is the development of a Bitcoin network listener (see Section 1.5.1). In fact, no Bitcoin deanonymization attack can be really complete without taking into account the *external information gathering* step, since this the step that allows to link anonymous clusters with real world identities. We plan to employ a listener to help in this crucial step. The undergoing work is presented in Section 4.2. We do note that we have also improved this gathering step by implementing a web crawler to look for Bitcoin addresses published on the Internet. Of course voluntary disclosure addresses are not always reliable but are especially useful to link highly visible merchants and services addresses publicly announced.

4.1 Other interesting transaction patterns

In this section we present our ongoing work about finding interesting transaction patterns in the Bitcoin blockchain.

We first remark the difference between looking for transaction patterns in the transaction graph or in the users graph. In fact, due to the clustering step, trans-

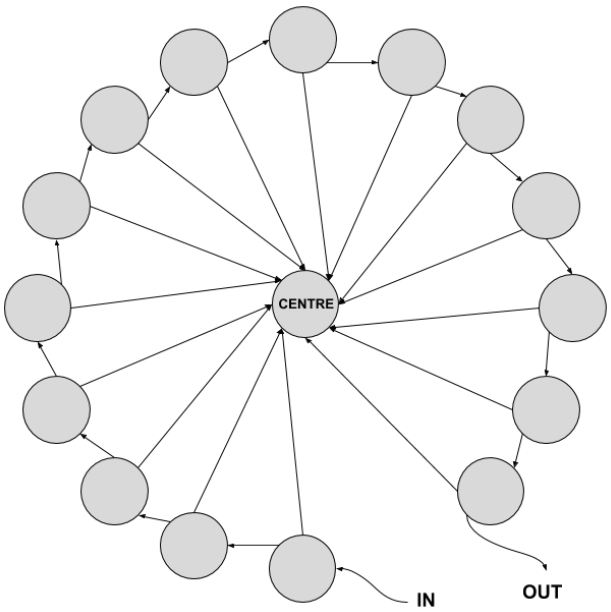


Figure 4.2: Wheel transaction pattern scheme

action patterns present in the transactions graph may disappear in the users graph (and the other way around). For example, we can consider a long chain, whose intermediate change addresses are later used all together as inputs of a single transaction. The chain will become a set of self loops in the resulting weighted directed hypergraph representing the users graph. Since it is interesting to recognize transaction patterns to potentially strengthen the change heuristic clustering rule, as explained in the previous section, we decided to identify transaction patterns on the transaction graph, rather than the users graph.

Among the different patterns we could find out, the present in this section the one we have studied the most so far, that we named *wheel*.

Definition 11 *Let A be the set of all addresses present in the blockchain, and T the set of all transactions. Given a set W of at least two transactions, each modelled as a tuple $(In, Out, InAmount, Fees)$, where:*

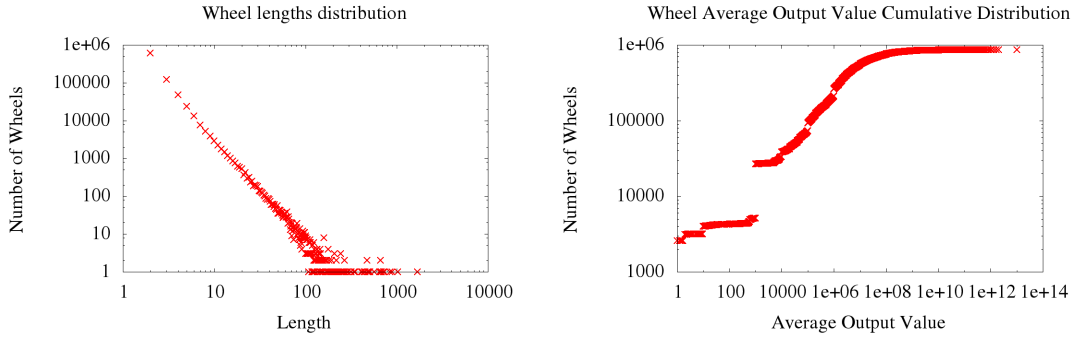
- $In \subseteq A$;
- *Out is a multiset of couples (o, b) where $o \in A$ and $b \in \mathbb{R}$, where b corresponds to the amount paid to address o by In;*
- $InAmount \in \mathbb{R}$;
- $Fees \in \mathbb{R}$;

we say that W is a wheel *of* length $n = |W|$*, if there exists an address $a \in A$, called the* wheel centre*, so that W satisfies the following properties:*

- $\forall$ *transactions* $t \in W$, *where* $t = (In_t, Out_t, InAmount_t, Fees_t)$, *holds:*
  - $|In| = 1$;
  - $|Out| = 2$;
  - *given* $|Out| = \{(o_1, b_1), (o_2, b_2)\}$ *then either* $o_1 = a$ $o_2 \neq a$ *or* $o_1 \neq a$ $o_2 = a$ *i.e. one and only one output is the wheel centre. We call the only output that is not the wheel centre* change *address.*
- *there exists an ordering of elements of* W *i.e.* $W = \{t_1, t_2, ..., t_n\}$ *so that* $\forall i$ *s.t.* $1 \leq i \leq n-1$ *the unique input address of* t_{i+1} *is the change address of* t_i.

In other words a wheel is a chain of transactions, linked one to the other, where the transaction at each step has exactly one input and two outputs, and one of the two outputs is directed to the wheel centre, while the other is used as next step in the chain. A graphic representation of a wheel is given in Figure 4.2.

By scanning our dataset looking for maximal transaction chains satisfying Definition 11, we found 860 172 wheels, containing a total of 2 546 785 transactions, i.e. 2.56% of all transactions in our dataset. Some measures of the wheels found are shown in Figure 4.3, where we show the distribution of wheel lengths (Figure 4.3(a)) and the cumulative distribution of average amount paid to the centre at each step of the wheel (Figure 4.3(b)).



(a) (b)

Figure 4.3: Wheel lengths frequency distributions (a) and cumulative frequency distribution of wheel average amount paid to the centre (b).

We remark that short chains of transactions are more likely to be classified as wheels. Moreover our definition allows for the same addresses to appear on the wheel circle (i.e. not the centre) multiple times, potentially even the extreme case where the circle always employs the same address in a series of self loops. We did notice this behaviour among our wheels, in particular by the *DeepBit* mining pool. In fact *DeepBit* always took part in peculiar wheels where only one address is used

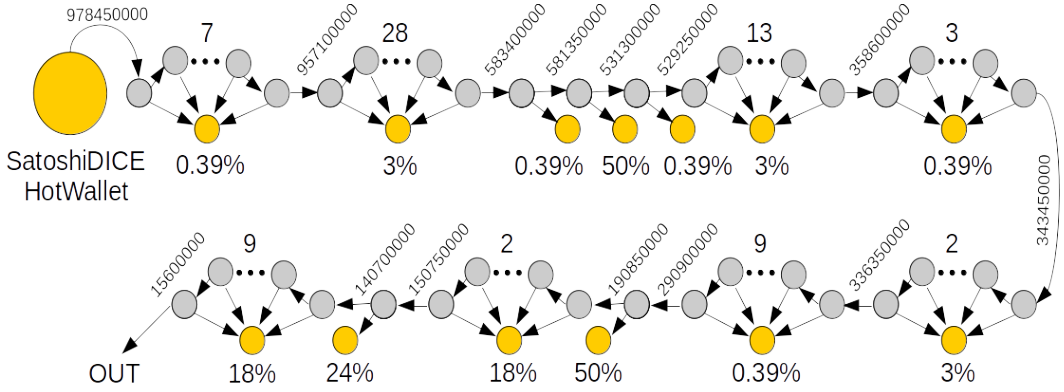


Figure 4.4: Interesting chain of wheels with all centres belonging to the SatoshiDICE betting service [30]. In the figure orange coloured nodes belong to SatoshiDICE and each betting addresses used as centre is labeled with its chance of winning. For each wheel is noted its length, i.e. the number of transactions it is composed of, as well as the starting and ending amounts.

both in the circle and the centre, hence collapsing in a single node with $2n$ self loops, where n is the length of the wheel.

To try and clean the dataset from short (and so less meaningful) wheels and these peculiar wheels, we decided to filter these patterns by only considering wheels with at least five different nodes on the circle, finding 121 015 wheels satisfying this property. By looking for known addresses, i.e. addresses of identified entities present in the tag datasets provided by [254, 108], we found 85 605 wheels with at least one tagged address. Moreover, 77 745 of such wheels (i.e. 64.24% of all wheels with at least five unique nodes on the circle) contained at least one tagged address related to SatoshiDICE [30]. By inspecting those wheels we found out a precise pattern. A SatoshiDICE betting address was the centre of the wheel, receiving bets from the addresses on the circle. Furthermore most of such wheels were connected between each other, i.e. the output of a wheel provided the input for another one. For example we show in Figure 4.4 a long chain of transactions that starts from a payout from *SatoshiDICE hot wallets* (i.e. addresses known to pay out the winnings) and keeps betting to a SatoshiDICE address (but not always the same). This chain contains multiple wheels, that are naturally spawn when the betting address remain the same, and so can be considered the centre of a new wheel.

We are still investigating the possible meaning of wheels, but it seems clear that wheels are very often used as betting pattern. This gives a strong hint about the artificiality of the wheel transactions, since it is very likely that the same entity has control over all the addresses on the wheel circle, and uses them only to keep betting.

We do remark how betting services in general and SatoshiDICE in particular have

been accused of being used as miking services [32]. So wheels could be potentially used as an obfuscation technique (see Section 1.5.2).

4.2 Bitcoin network listener

Aim of a Bitcoin network listener is to gather the live data travelling on the Bitcoin P2P network. This step requires the deployment of one or more listeners nodes in the network. As we have seen in section 1.5.1 proposals to deanonymize users using data gathered at network level already exist in the literature. The authors proposal in [22] is really invasive, requiring a heavy polluting of other nodes connections lists, thus it can lead to a significant DDoS on the network. We want to observe the network influencing it as little as possible, so we deem this technique inappropriate for a real world scenario. The other proposal we saw, [158], is not applicable as well because it relies on anomalous relaying behaviour and so is not generally applicable. During our work we decided to use only non invasive and lightweight techniques, gaining less data (both in quantity and quality) but perturbing the network as little as possible.

We do note that the data gathered from the listener allow us to also study other open problems in Bitcoin, such as the fraudulent mining techniques and double spending attacks we have seen in sections 1.3 and 1.4. In particular we refer to the increased possibilities offered from the live network data collection, as shown in section 1.5.1 with the malleability attack detection example [74].

A very interesting application of the listener data, only indirectly linked to anonymity, is the P2P network topology discovery. Knowing the topology of the network is useful for a deanonymization attack, due to the assumption that the first node to broadcast a transaction is also considered to be its creator (see Section 1.5.1), but it also has other advantages. For example we could use the topology data to perform graph structure analysis (e.g. check if is it a small world or a random graph) of the live network and evaluate network metrics (e.g. node stability, diameter, influential nodes discovery, etcetera). All those information are very useful to study the information propagation in the network, allowing us to see if the actual propagation respects the theoretic models. From an attacker point of view knowing the network topology allows the study of connectivity, finding a small subset of nodes to target to maximize the chances of success during sybil attacks, alternative reality attacks or DDoS attacks. A related interesting application of topology discovery is trying to find mining pools gateways, possible, for example, employing techniques derived from [187]. To find gateway nodes is very interesting from a decentralization point of view because these nodes should be well hidden, being the most valuable targets for DDoS attacks. At the best of our knowledge there are no up to date deep topology analysis of the Bitcoin live network. The last relevant results were presented in [187] but the technique used to obtain them is not applicable any more because the timestamps policy on which is based was changed with version 0.10.1

of the official Bitcoin client, as explained in section 1.5.1.

During my first year of PhD studies I had the great opportunity of spending part of the PhD period abroad at the *Research Group for Distributed Computing* (*DISCO*) which is part of the *Computer Engineering and Networks Laboratory* at *ETH Zürich* under the supervision of professor Roger Wattenhofer. During this period we started working on the implementation of a Bitcoin network listener. We kept working on the project in Pisa and a detailed description of the developed client and some experimental results obtained with it are presented in Appendix A.

II

Blockchain Based Access Control Systems

Introduction

We have explained in this thesis introduction how blockchain technology can be used in many different fields of application outside of its original intended one, i.e. cryptocurrencies, shown in Part I. Especially in the last few years (i.e. after 2015) proposals of new blockchain applications have boomed. Surfing this trend we decided to give our contribution by looking into possible blockchain applications of our own. Despite the great effort in applying blockchain technology to anything, still, when we started our work, there were no proposals to apply it to a field that our colleagues had been working on for some time: Access Control systems. This seemed surprising since using a blockchain could lead to some clear advantages in such a scenario, most of all improving transparency of the decision process. Deeming the topic of Blockchain integration in traditional Access Control systems worth of studying we oriented our research efforts to such task. In the rest of this part we will show the resulting proposals and results.

First in Chapter 5 we present some of the most promising non-cryptocurrency applications of blockchain technology. Then in Chapter 6 we provide the reader the needed background in Access Control systems and the related standards we used in our work. We also discuss the possible advantages that the integration with blockchain technology could provide and present the few related works available.

In Chapter 7 we show our work in this field. As a first step to test the possibility of integrating Access Control systems with blockchain technology we decided to use the blockchain to manage only Access Control policies (i.e. documents stating the set of rules expressing the rights of subjects to access resources and evaluated at access request time against the current access context, see the introduction of Chapter 6). So we propose a pluggable system to publish, update and revoke policies on the blockchain. In traditional Access Control systems policies are managed by a dedicated component (see Chapter 6) that stores them either locally or on external services. So, our blockchain policies management component can be plugged in traditional systems without any change required from them, while, at the same time, granting some new properties. Since policies are published (and updated) on the blockchain, they are visible to the subjects of the scenario, consequently any user can know at any time the policy paired with a resource and the subjects who currently have the rights to access the resource. This approach allows distributed auditability, making evident when a party is fraudulently denying the rights granted by an enforceable policy.

Furthermore we explored the possible novel applications of such a policy management system when paired with a blockchain supporting a cryptocurrency. In fact the native concept of value provided by such a scenario allows for the decen-

tralised transfer of access rights among users, without any third party intervention, including the resource owner. This possibility opens completely new possibilities and application scenarios.

To test our proposals we implemented a proof of concept implementation based on the most widespread Access Control standard, i.e. XACML (see Section 6.1), and the most famous cryptocurrency blockchain protocol, i.e. Bitcoin (see Section 1.2).

Chapter 8 presents the main contribution of Part II. In this chapter we show our design proposal for an access control service based on blockchain technology. Differently from Chapter 7, the proposal outlines a fully fleshed out Access Control system with different levels of integration with the underlying blockchain, including a fully blockchain system where the protected resources are smart contracts on the blockchain. One of the main contributions of the work is to delegate to the blockchain not only the policy management (as in Chapter 7) but the decision process as well. The result is that an entire Access Control system can be deployed and executed by the blockchain. Since we need powerful enough tools on the blockchain to express and execute the Access Control logic we need to use smart contracts, and consequently a smart contract supporting blockchain. In fact the key idea behind our approach is to codify Access Control policies as executable smart contracts, hence obtaining decentralised self evaluating policies. The decentralised and trustless nature of the blockchain and smart contracts execution enables the subjects requesting access to the resources to verify that the policy has been correctly enforced, disclosing malicious or faulty third parties fraudulently denying access to subjects.

To study the feasibility of our proposal we present a working reference implementation based on the XACML standard as in Chapter 7, but designed for the Ethereum protocol, due to its native support for smart contracts. We have not only analysed the theoretic expected advantages and disadvantages of the proposal, but also evaluated its usability in practice. In fact, we have measured the main performances of our system through experiments on two different real world Ethereum testnets.

Finally in Chapter 9 we present our conclusions and show the directions we are taking in our current research. As such we present a possible integration to the system presented in Chapter 8.

5

Blockchain 3.0 Applications

Abstract

In this chapter we present a sample of interesting Blockchain 3.0 applications, i.e. applications of blockchain technology not related to cryptocurrencies. In particular we study end-to-end verifiable electronic voting, healthcare records management, identity management systems, decentralized notary, intellectual property protection and supply chains management. For each topic we first explain the scenario and requirements. We then present how blockchain technology is applicable and what advantages it might bring. Finally we present both academic and commercial systems, either proposed or already implemented.

Blockchain technology was first proposed to support cryptocurrencies like Bitcoin, so cryptocurrency blockchains and related applications are often labelled as *Blockchain 1.0*. The main achievement of cryptocurrencies is the decentralisation of value transfers between untrusted entities, but many other more complex applications can be built on top of this disruptive innovation. The introduction of smart contracts to realize decentralised applications (*Dapps*), decentralised autonomous organizations (*DAOs*), smart property, smart tokens, etcetera paved the way to automated financial applications based on cryptocurrencies. All these novel applications in the financial area made possible by the union of smart contracts with digital currencies are labelled *Blockchain 2.0*. But, as we have explained in this thesis introduction, blockchains are not limited to cryptocurrencies, they are just a possible implementation of the broader concept of distributed ledger, which may contain arbitrary information, not necessarly related to money of finance. All applications of blockchain technology referable to the wider spectrum of non cryptocurrency-related distributed ledger uses are commonly referred as *Blockchain 3.0* applications. We do note that, even if such applications are conceptually independent from cryptocurrencies, they can still benefit from an integration with them, and in practice are often deployed on a cryptocurrency blockchain such as Bitcoin or Ethereum.Blockchain 3.0 means bringing all the useful properties obtained by the blockchain trustless decentralization (such as immutability, transparency and no need for intermediaries) to traditionally centralized systems.

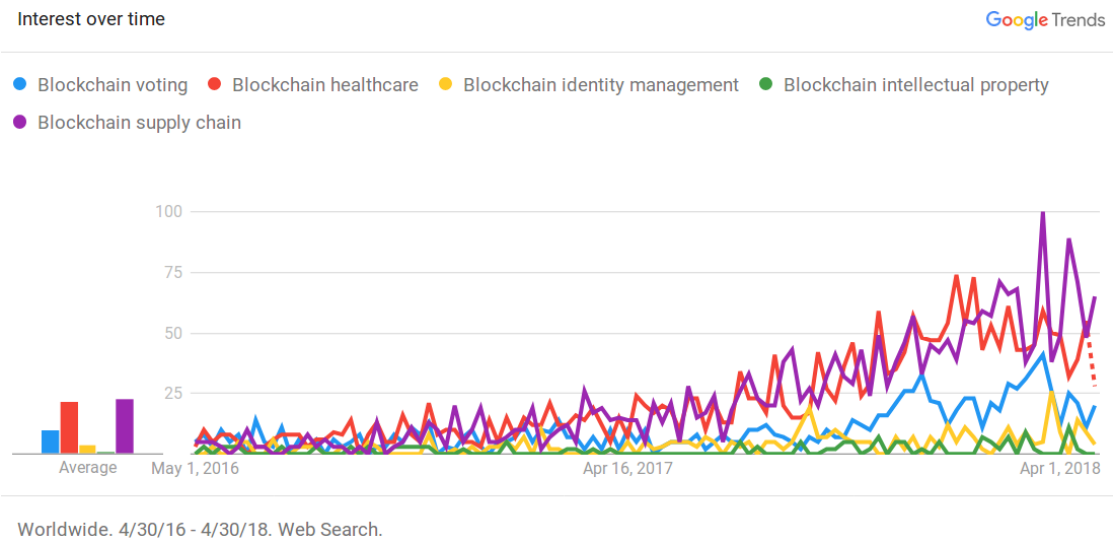


Figure 5.1: Google trends [126] chart from the 30th of April 2016 to the 30th of April 2018 for the search terms *Blockchain voting* (blue), *Blockchain healthcare* (red), *Blockchain identity management* (yellow), *Blockchain intellectual property* (green) and *Blockchain supply chain* (purple).

Due to the recent hype in blockchain technology, during the last two years there have been new proposals for Blockchain 3.0 applications almost everyday, as every company feels the need for a blockchain. The Google trend chart for the terms of the topics presented in this chapter are shown in Figure 5.1, to estimate the relative interest in each of them. But we should remind that the nice properties of a blockchains are often gained at the expenses of scalability and resource costs. Only scenarios where the benefits clearly outweigh the drawbacks will see the disruptive possibilities of blockchain adoption. In the following sections of this chapter we present the main applications where the use of blockchain technology seems more promising.

5.1 Electronic Voting

In this section we present the possible application of blockchain technology to *electronic voting* (or *e-voting*) systems. We remark that e-voting refers to a very specific application, i.e. we define electronic voting any type of voting system where votes are cast and tallied through electronic systems [3]. This is in general not the same as *distributed* or *digital voting*. While distributed voting is a general enough process to refer to a broad array of applications (for example leader election between autonomous agents), e-voting was born with the precise use case of human political elections in mind. In this framework, e-voting may apply to many different voting

schemes, not only related to political elections (that may have different rules themselves depending on the country), but also, for instance, to stock holders voting on corporate decisions.

Problem Definition. E-voting presents advantages and drawbacks compared to traditional voting systems. For example, a usually mentioned advantage is the greater turnout expected [264], especially for e-voting systems allowing remote voting (i.e. the ability to vote without having to physically go to a designated voting booth). For a voter expressing a preference through few clicks on a digital device without leaving its home might save time and money, easing the election burden (both as time consumption and monetary cost) on the single user as well as the entire society. It might make voting especially easier with voters with special needs or reduced mobility, as well as voters travelling abroad or busy working. This is even more true if the voting procedure is repeated often, like for direct democracy initiatives. One of the main disadvantages is the increase risk for election tampering, since by basing the voting system on electronic devices we expose it to hacks of such devices. Moreover the risk-reward trade-off of an attacker is higher, since the hack attempts require less resources and can be conducted on a much greater scale than traditional physical systems. Furthermore the proposed systems might look more complicated than traditional ones, for example if a system requires public key cryptography it usually entrusts the voter with the security of a private key, and most users may compromise their security by not protecting such secret information adequately. This also opens a new attack venue for an adversary that could try to steal such private information from the users.

E-voting requires the satisfaction of some precise properties derived from the original human use case, which mainly concern security, auditability and privacy and, when satisfied, allow a so called end-to-end (E2E) verifiable voting system [56]. By security and auditability we mean that it is not possible to tamper with a vote (i.e. changing the preference expressed by the voter) without the voting entity ability to notice that. Mainly the voter should be able to check that each vote was cast as intended, recorded as cast and tallied as recorded. Any outside observer should also be able to verify the election result (i.e. all and only allowed votes were considered and tallied correctly) without having been involved in the election. Furthermore we could desire that the voter should be able to prove to a third party authority that those checks failed (and so the its vote was tampered with) without sharing any private information about its vote. By privacy we mean that each vote should be only known to the voter even while voting. A stronger voter privacy property called coercion resistance might also be required, which means that the voter could pretend to cooperate with a malicious entity while instead voting its own choice.

Usually E2E voting is achieved by creating a receipt for the voter representing its vote in an obfuscated way. This receipt is used alongside some public data stored on a bulletin board, that, coupled with the private information of the receipt, should provide auditability. No information should be leaked from the bulletin board nor

the receipt about each vote, the only source of information should be the result itself. Finally we note that no security assumptions (i.e. not to be malicious or faulty) are bestowed on the voter, the voting devices or the election authority, they are all considered untrusted by each other.

Blockchain Based Proposals. Curiously enough, the first proposal of an e-voting system is from Chaum in 1981 [56], the same author of the first digital currency scheme two years later [55].

Several proposals have emerged on how to implement an e-voting system on top of most popular blockchains available today. Of course the easier platform to achieve such goal is a Bitcoin-style cryptocurrency oriented blockchain. In the following we show the outline of a possible implementation of an e-voting system in such a scenario, based on real existing proposals. In general the voting process takes place in three conceptually separated steps: voters registration, vote casting and votes tallying.

At voter registration time, the system needs to enforce voter eligibility, i.e. verify that all allowed voters are allowed to cast a single vote and no one else is. The general scenario includes a centralized trusted authority to recognize allowed voters. The simplest solution to achieve this is for voters to create a new address (i.e. cryptographic key pair) and advertise it to the authority. The authority can then send a token payment to the address from one of its publicly known addresses, or publish the voter address in an eligible voters public list. At the same time, each candidate advertises a set of addresses representing themselves (of course it is in its own interest not to advertise addresses it actually does not control). To cast a preference each voter sends its received voting token (or a freshly created token in case of eligibility list) to an address of the chosen candidate. The tally can then be done by simply counting all the tokens received by each candidate set of addresses.

This simple framework is not complex enough to guarantee E2E voting security, but is helpful to easily show advantages and drawbacks of using a public blockchain. First of all we note that using a public blockchain trivially protects the system from a malevolent election authority. Since the authority has no control neither at casting nor at tallying time it cannot tamper with the election. This can be weakened in systems relying on a permissioned blockchain, which still rely on a trusted entity to evaluate voter eligibility, but this is inevitable in traditional systems and still auditable in case of unduly denial of voting rights or fake voters forging. A voter can prove that they were not included in the eligible list or show that its address has not received any voting token (but at the price of disclosing which address belongs to it). The process should be of course secure enough to guarantee that the eligibility authority cannot fraudulently link voters with bogus addresses they control (for example by requiring an unforgeable voter confirmation step). On the other side an external observer can chose a set of addresses that voted and challenge the authority to show that those addresses really corresponded to rightful voters. Furthermore is important to note that it is needed to also employ obfuscation techniques to prevent

a direct linking between voting address and voter identity in the eligibility authority table, otherwise the authority would be able to see what candidate each voter choose, breaking the fundamental vote secrecy property. Another important issue of such systems is that it allows for real time tallying, which is, is in general, a non desired property, since it can influence the election outcome, and, more seriously, it can leak vote private information if a precise enough timing analysis is possible.

If we compare this simple system with the general E2E voting approach described in the previous section we clearly see how blockchain technology is used as the bulletin board of the system. This is the main conclusion reached in [196]. Not only a blockchain satisfies all the properties required from a secure bulletin board but it also introduces new useful properties. As already stated, a public blockchain decentralises the bulletin board management and control hence protecting the e-voting infrastructure from an untrusted central authority. This would also contribute to system robustness and availability since it would rely on the resilience of the underlying blockchain. On the other hand, voting would require to create transactions and so pay fees in a public blockchain. This could harm usability and vote accessibility.

Several proposal have been advanced to use the novel smart contract capabilities offered by recent blockchains. There is no common implementation to outline here but they all rely on the principle of delegating the aforementioned operations of voter authentication, votes casting and tallying to smart contracts. This allows to have smart ballots and smart tallying contracts, possibly able to enrich the system with new functionalities. We do remark that this introduces further problems of scalability depending on the blockchain chosen. In fact, if the smart contracts are required to perform costly cryptographic operations they could become too slow and too costly in practice for an usable system supporting an high number of users. For example the *Open Vote Network* smart contract based implementation proposed in [178] can support at most sixty voters. With today price rise of Ether (block of Ethereum main chain 52 971 00) the proposed system would cost more than 4 800 USD for one single election among sixty voters, i.e. abut 80 USD per voter (even if in practice higher gas price should be offered to ensure that such demanding transactions would be executed in a timely fashion).

Cases Studied. In the following we summarize the main proposals both commercial and academic that we could find in the available literature. Do note that due to the novelty of blockchain technology almost all of the existing systems have been proposed in the last two years (i.e. since 2016). The most cited commercial proposal is BitCongress [33], even if the system seems to have been since discontinued. It used Bitcoin coloured tokens through Counterparty [64] to authenticate voters and cast votes, and Ethereum smart contracts to tally votes. Each voter was identified in the system with only one vote associated to it during its lifetime, unfortunately this prevents a voter from taking part in multiple elections at the same time. Another famous proposal was *FollowMyVote* [116, 98] from a no-profit organization. The project is open source and based on BitShares [36], a fork of the Bitcoin pro-

tocol. The underlying not so popular blockchain makes it more vulnerable and less robust. The system allows for three different election types: Proportional Representation, Mixed Member and Majority. It still requires a central eligibility authority, with blind signatures to obtain voter identity obfuscation. Other proposals using Ethereum include *Procivis* [217], the aforementioned Open Vote Network implementation [178], *Democracy.earth* [75], and *Polys* [212] over a private Ethereum fork. Proposals to use a blockchain with traditional voting booth electronic devices are *VoteWatcher* [271] and *VoteBook*[155]. Other projects include blockchain agnostic *Secure.vote* [239], commercial *Votem* [270] and *TIVI* [260], academic *VOLT* project, Spanish *AgoraVoting* over Bitcoin [1] and *Inno.vote* over the *BallotChain* blockchain [143]. Unfortunately all this projects have no E2E voting security formal proof.

Only a few formal academic proposals have been presented (e.g. *Remotegrity* [283]), we will show them in temporal order in the following. The first such proposal was [285], based on the scheme of a distributed lottery and using zero knowledge proofs [112] over the Bitcoin blockchain. The next work [165] proposed another approach using a trusted third party to manage the eligibility process. The proposal is applicable either to the Bitcoin blockchain or any permissioned blockchain and follows the same general scheme outlined before about voting schemes in Bitcoin. A similar proposal is shown in [67], using blind signatures alongside the Bitcoin protocol. A weakness of the proposal is the requirement for prepaid Bitcoin cards to be given to the voters. A similar scheme was used in [255], but based on Zerocoin to enhance voters privacy. Similar approach is used in [256], that uses Zcash instead to anonymize transactions. After the aforementioned proposal on Ethereum using smart contracts [178], in [15] the authors propose a new e-voting system based on a Shamirs secret sharing scheme built over multisignature PayToScriptHash scripts on the Bitcoin blockchain. They also present an enhancement of the CoinsShuffle [235] technique, called *CircleShuffle* to further decouple the inputs of a CoinShuffle transaction from each output. Finally another proposal on Bitcoin following the usual scheme was presented in [24]. Nevertheless it does not satisfies all the E2E secure properties and the authors suggestion to use a permissioned blockchain to alone solve the issue might not be enough.

A final remark. As we have seen in the previous sections, Blockchain technology in general already employs an idea of distributed *voting* (not properly e-voting) to achieve a consensus on the next block to be appended to the chain. Most distributed consensus algorithms proposed can be seen as distributed voting algorithms to chose a miner to add the next block to the chain. In those systems miners have different voting weight according to their commitment to the election. The commitment is expressed in different ways depending on the consensus algorithm chosen, for example with PoW a miner shows commitment by dedicating computational power, while with PoS it shows funds ownership. The main difference with respect to real voting systems is that the election is probabilistic, i.e. each candidate has a probability to win proportional to the number of votes he receives, but there is no

certainty that the candidate with the most votes will win. Nevertheless on the long run the system will converge to reflect the voters will through the number of blocks mined by each candidate (even if this can be further complicated by the dynamic nature of miners that can freely enter and leave the system). This is why in practice the consensus algorithm has been used as a tool to vote on important decisions for existing blockchain protocols. For example the Bitcoin community has often employed this trick to decide on protocol upgrades. Miners are asked to cast a vote about the upgrade as data in their coinbase transaction. This allows to have an approximation of the mining community will once enough blocks have been mined. This approach feels natural for a cryptocurrency such as Bitcoin where a protocol upgrade could require an hard fork that might not be accepted by all the users and may cause an harmful split in the network. Casting votes in real blocks allows to check if the vast majority of miners really contributing to the network at the time do agree with the proposal.

5.2 Health Care

Problem Description. In the digital era more and more private information about ourselves is stored and managed electronically. That information needs to be properly protected from theft and unauthorized uses. One particularly sensitive information is medical data: each medical examination produces valuable sensitive data belonging to the patient that needs to be shared with their doctors, pharmacies or insurance companies, but, at the same time, protected from external access. Nowadays most national health systems are trying to collect all personal medical information of a patient inside a unique electronic medical record. Such a record contains very sensible information produced during an entire lifetime, and yet it is often managed by medical institutions and practitioners not technically aware or equipped enough to guarantee the appropriate security levels. Moreover, most medical institutions store and create patients' medical records in different formats often not compatible not only between different nations, but sometimes even between different labs inside the same hospital.

The need for a system to manage and store personal records in a guaranteed secure way has lead to a lot of proposals to use a blockchain. As cryptocurrencies have made users the direct owners and managers of their own funds, so blockchain managed electronic medical records would make the patient the real manager of their sensitive data.

Blockchain Based Proposals. Despite the many proposals, most follow a common general idea: a system storing medical personal records on a blockchain under the control of their owner (see Figure 5.2). Through the cryptographic security (digital signatures, etc.) of the blockchain the user would be the only one able to grant access to their own records [282, 148]. Storing the entire records on the blockchain would not be a good idea for two main reasons. First, it raises obvious privacy con-

cerns since all data in the blockchain would be visible to all other users. Of course the data could be obfuscated or encrypted, but this could still not be enough from a legal point of view. Moreover, the public and immutable nature of the blockchain would mean that the encrypted data would remain forever visible on the chain. If a future technological advance would make that data no longer secure then its content could be retroactively read. The second reason is that the record could become very big in size. The medical data contained inside the record is by its own nature space demanding, because it could comprehend a lot of images (for example from MRIs) and because it keeps growing as the patient ages. The medical record of a lifetime would take a lot of memory to be stored. On the other hand, blockchain space is scarce due to its decentralised nature, and the entire blockchain with all its data should be replicated on each node, so each bit it contains consumes storage space for each node. It would also negatively impact the performance of the communication network, since big blocks to store a lot of new data would require a lot of bandwidth to be relayed among nodes, delaying new blocks discovery notifications and so potentially increasing natural forks probability (which are causes of wasted mining resources for the entire network). This means that it is in practice unfeasible and not desirable to store entire medical records on the blockchain. Luckily this is not necessary, but, luckily, the system works all the same by simply keeping on chain only pointers to where the records are actually securely stored. This is the same approach proposed in [286] to manage personal information through a blockchain. In those systems the blockchain acts only as a decentralised secure system where users can manage the access to their personal records [167, 50, 210] that despite a slightly different approach follows the same base principle. The real data is stored off-chain in a traditional manner, of course with care to preserve the privacy of the content.

In such a system, users alone are in control of their own data by granting or denying access (e.g. to pharmacies or insurance companies) to it. Of course the users don't have to grant access to their entire medical record, they can choose what data to share with each subject. At any time the users would be aware of who has access to which part of their own data, since they are the only ones able to grant such access. In such a system there would be no intermediaries and no need for trusted third parties. If this system is deployed on top of a blockchain with a native currency, such as Bitcoin or Ethereum, then the users could also be directly paid in exchange for grated accesses, for instance, if the user grants access to their data to a scientific organization for research purpose. Furthermore, since users are in charge of securing their own data, it would relieve both medical institutions and companies from this difficult and costly task. In turn, this would have an impact on the corresponding legal framework, since users are the ones giving consent for the use of their data each time they grant an access. At the same time, the notary nature of the blockchain (due to its immutability and timestamping it automatically constitutes an audit trail) would increase the system transparency and auditabilty. A patient,

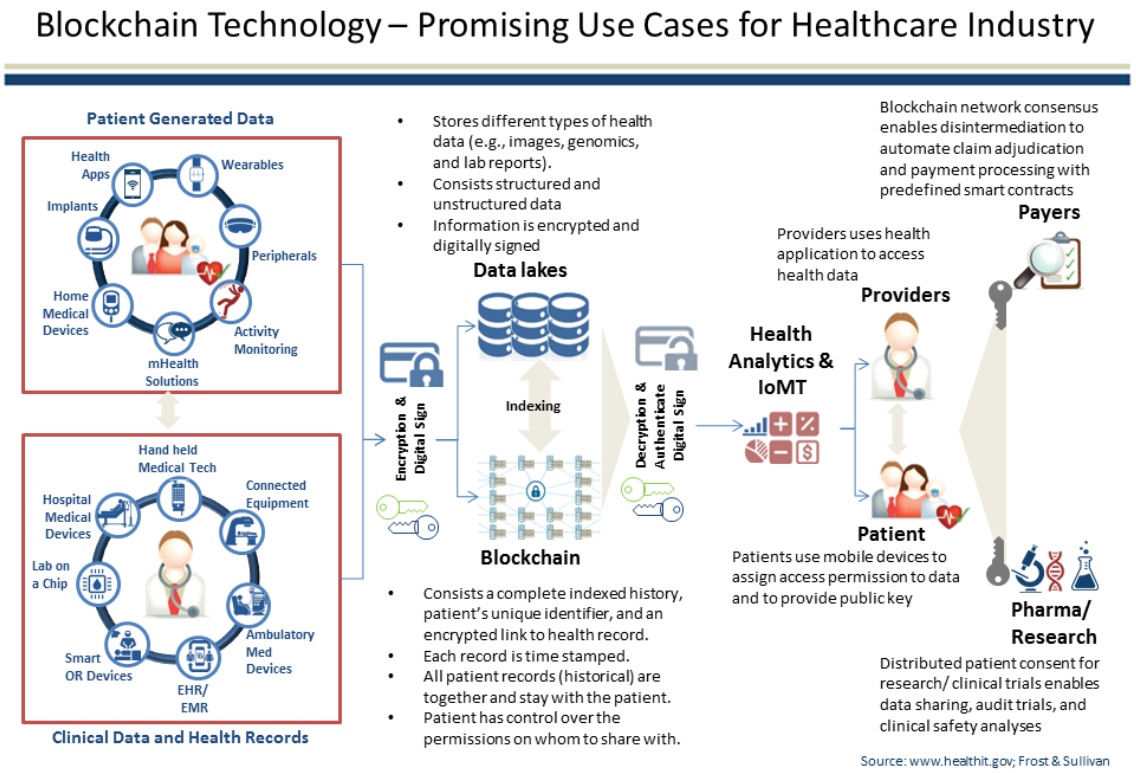


Figure 5.2: Blockchain technology application to healthcare.

for example, could prove to have taken a certain test without disclosing personal information or relying on a third party. This would cut expenses for the patient and greatly shorten the time required compared to current systems. Furthermore, it could also help detect health care frauds. Finally, we note that cryptographically advanced solutions could be employed to allow use of the private data without actual data disclosure (e.g. homomorphic encryption computations [122]).

Another advantage of such a system would be the universality of the record format. The blockchain would act as an intermediary that all the different systems have to interact with, dictating a common language to exchange data. This could ease the lack of format interoperability problem often experienced nowadays with personal medical records [284]. The existence of a single point of access for medical information in a single format could also benefit health studies and research. Users could grant access to their medical data for aggregated medical studies, that would benefit from a huge patients base to analyse (potentially in an automated way, e.g. with big data techniques [162]). Users could then be directly rewarded for their participation by blockchain micropayments.

In practice we do not expect a single blockchain to manage the global population

medical records, it would be more feasible to have different blockchains at different institutional levels (e.g. from national level to single medical institution level). All those blockchains could be federated to allow interoperability without the need for data replication between one another. We also do not expect all these blockchains to be public. A permissioned solution with medical institutions as nodes would seem like a natural implementation of the system [93]. Furthermore, employing blockchain technologies with smart contracts support would also enable more expressive systems.

Cases Studied. One proposal using smart contracts (on the Ethereum network) is *MedRec* [96], one of the few working academic proof of concept implementation available in the literature. The system follows the basic scheme outlined before, but also introduces an interesting twist. It introduces access to aggregated and anonymized data (usable for example for research) as mining reward to foster participation in the expensive mining process.

Very few proposals have also been presented to apply distributed ledger technology to epidemics relief. Of course blockchain technology through cryptocurrencies can already contribute by allowing direct micro payments to make fast donations. In [61] the author proposes to use a distributed ledger to monitor a disease and its spreading. The concept can be strengthened by coupling the idea with smart contracts capabilities to trigger automatic responses or alarms in case of preconditions concerning the monitoring of epidemics.

Outside of the academic world, many companies have tried to apply blockchain technology to the health care sector. Blockchain technology has even been used in practice to protect electronic medical records in a famous experiment by Estonia [133, 184, 7]. Among the commercial proposals joining blockchain technology with some aspect of health care we remember *Gem* [121], *Hashed Health* [130], *SimplyVital Health* [243], *PokitDot* [211], *Robomed Network* [228] and *Healthcare Working Group* of the Hyperledger project [131].

5.3 Identity Management Systems

Problem Definition. An identity management system is used to identify an entity in a digital system and all data and means of authentication needed to recognize that entity. Basically, such a system labels each entity with an identifier (usually in a human friendly format, e.g. a meaningful string), it provides a way for the entity to authenticate (often by proving knowledge of some private information, e.g. a password) and stores its relevant identity information. Identity data can be as simple as name, age, date and place of birth, etcetera for an electronic passport, or as complex as bank and credit data for a financial application.

Nowadays identity management systems are mostly centralised and isolated between each other (see Figure 5.3 (a)). Users are forced to rely on a different central service to manage its identity data in each different domain. Not only it is ineffi-

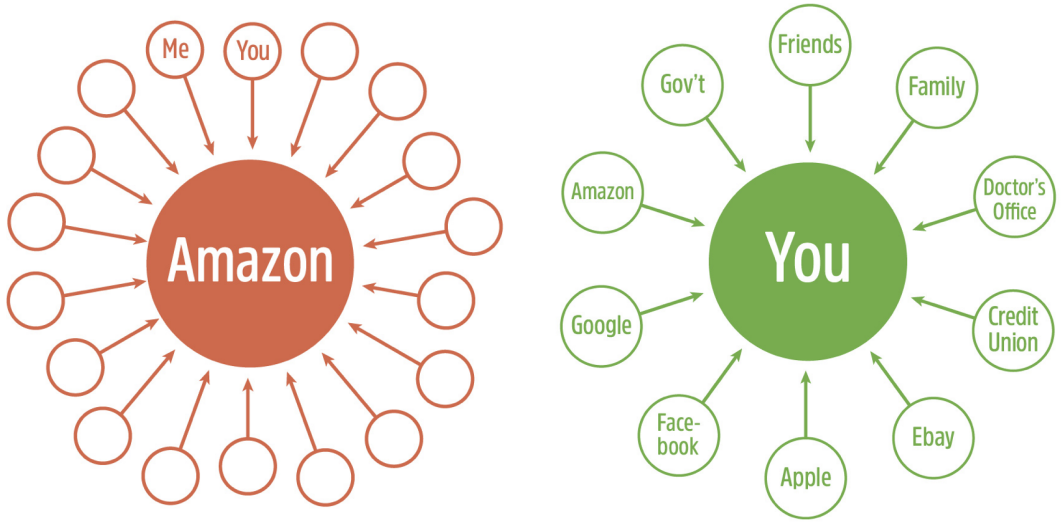


Figure 5.3: Traditional centralised (a) and self-sovereign (b) identity management system schemes. Source: [4].

cient and cumbersome for users (forcing them to remember a lot of different private authentication information) but also dangerous for their privacy. Central systems need to be trusted with sensitive data by users not only not to maliciously exploit such information, but also to effectively protect it from external attacks, considering also that big quantities of valuable data stored all together in a single place are attractive beacons for hackers. To improve user experience, federated identity systems have been proposed [259, 261], where identity managers remain centralised for each domain but users can use the identity of a single domain to access all the federated ones. The identity portability between systems can extend beyond authentication also allowing for some identity data to be shared. Even if this solution eases the burden on users, it still gives them no control over their identity data that remain centralised for each domain as before. A *user-centric* identity system [53], instead, would solve privacy issues by putting the user in charge of its own identity data, not third parties. Basically a user-centric identity system is a federated system where the identity control is centralised in the hands of the user. Its natural implementation is to use a blockchain (or any other distributed ledger implementation) to obtain a *self-sovereign* identity system [259, 261], where the identity system is agnostic of the underlying applications using it (see Figure 5.3 (b)).

Blockchain Based Proposals. The idea of a self-sovereign identity system based on blockchain is no different from the idea of blockchain based personal health record shown in Section 5.2. If we consider health related data as identity information the systems are interchangeable. For example in [88] a DLT based self-sovereign identity systems is described as allowing entities to create immutable *identity records*

represented as *identity containers* able to *accept attributes or credentials from any number of organizations [...]. Each organization can decide whether to trust credentials in the container based on which organization verified or attested to them.* This is exactly the same concept behind blockchain based health records and, as such, they share most of the advantages. If we analyse the desired properties required by such a system stated in [259], we notice how effortlessly they can be satisfied by a public blockchain based implementation:

- Control. *Users must control their identities.*

  Access. *Users must have access to their own data.*

  Consent. *Users must agree to the use of their identity.*

  A blockchain is censorship resistant, users not only are free to join independently from any third party, but are also the only ones controlling their own data, they can access and update it without intermediaries. Since users alone control their identity data, they alone decide to whom to grant access to it.

- Minimalization. *Disclosure of claims must be minimized.* This can be achieved by storing the identity data in a secure way to provide users privacy. Of course no plain private data can be stored on a publicly accessible blockchain. As we have already explained when talking about health records in Section 5.2, advanced cryptographic techniques can be employed to verify some properties about the data without disclosing private information about it (e.g. proving that a user is older than eighteen years old without exposing their age).

- Existence. *Users must have an independent existence.* This is clearly true in a blockchain, that is independent from the applications using it.

- Transparency. *Systems and algorithms must be transparent.* This is guaranteed by the decentralised, open source and non-proprietary nature of a public blockchain.

- Persistence. *Identities must be long-lived.* Identities and their data would last as long as the underlying blockchain is not abandoned. The use of an immutable blockchain however could lead to issues to guarantee the opposite of this principle that is sometimes required: the *right to be forgotten.* A blockchain based system wishing to grant this property should be designed accordingly.

- Portability. *Information and services about identity must be transportable.* This is trivial using a blockchain that constitutes the only point of access for external services requests. The user remains the only one in control of its identity independently from the system requiring access to it.

- Interoperability. *Identities should be as widely usable as possible.* This is related to the previous bullet and is granted by using a blockchain as single hub for external systems to interact with.
- Protection. *The rights of users must be protected.* Guaranteed by the security and decentralisation of the underlying blockchain.

As pointed out in [261], those properties are all expression of the three main properties usually required from an identity system:

- **Security**, the identity information must be kept secure
- **Controllability**, the user must be in control of who can see and access their data
- **Portability**, the user must be able to use their identity data wherever they want and not be tied to a single provider

Besides the user direct control of its own data and the censorship resistant inclusiveness pointed out before, a decentralised blockchain could also lead to the practical advantage of reduced expenses. This is not only true for the users, also counting the potential costs of identity thefts and private data leaking of traditional centralised solutions, but also for the external services that would not have to store and protect any more private information. As a whole, private information would only be managed by the users, so there would not be the need to replicate it among the interested services with the related costs and privacy issues.

We also note how blockchain could in practice introduce novel issues for users, especially about the practicality of the system. For example, users would be alone in charge of the management of all the cryptographic keys protecting their identity information. In practice trusted escrow systems, possibly at a state level, could be necessary to recover lost keys or roll back mistakes made by users. For examples Id-cards issued by the government storing the corresponding keys could be employed, accepting the need for trust in the government issuing them.

Cases Studied. Several proposals and actual blockchain based identity management services have emerged during recent years. In [94] the authors identify *uPort* [268], *Sovrin* [261] and *ShoCard* [242] as the three most representative proposals. uPort implements a self-sovereign identity system through smart contracts on the Ethereum blockchain [267]. As such, external services can interact through other smart contracts. The identifier of a user is the Ethereum address (called *unique uPort identifier*) of its main identity manager contract, and the connected identity data is not stored directly on the blockchain but rather through the hash of the actual data that is stored on IPFS [146]. Sovrin, instead, builds its self-sovereign identity system on a public but permissioned ad hoc blockchain [249]. The trusted miners, called *stewards*, are controlled by the non-profit *Sovrin Foundation* to ensure their honesty. Even if data can be stored on chain, it is advised to store it locally

and disclose only to agreed parties through cryptographic side channels. Same as uPort, there exists a feature based on a list of trusted entities to recover an identity in case the corresponding secret key gets lost. Differently from the two previous examples, ShoCard is not used to build a self-sovereign identity system, instead, it relies on a centralised service that creates and stores user identities on a blockchain (based on Bitcoin [241]), alongside pre-existing identity certifications (e.g. driver's license). That information can later be linked and verified to prove entity identities. A user creates an initial transaction to start a new identity, and its identifier (called *ShoCardID*) is the hash of that transaction. Using that identifier as an anchor, additional identity information can be added to the user or verified. The need for a centralised service (the ShoCard server) makes the system actually centralised, since despite relying on a public blockchain it would be rendered unusable if the central server were to go down. This ShoCard and the other systems with similar paradigm (e.g. BitID [34], ID.me [141] and IdchainZ [140]) are unable to provide a real self-sovereign identity system. More proposals and actual services related to identity systems employing blockchain technology at some extent can be found in [88] and [149].

5.4 Decentralised Notary

The immutability and timestamping properties of a blockchain allow to store on it a piece of data at a certain time with the mathematics guarantee that it will not be modified in the future. This leads naturally to the idea of using the blockchain as a decentralised notary system. It means that users can submit any information to the blockchain, and then, through the distributed consensus, the miners community will add this information to a block at a certain time. Since it would then be unfeasible to modify such a block in the future (under blockchain security assumptions), it is possible to prove that at that time that piece of information existed and was not tampered with. If the information stored represents the hash of an electronic document then it is possible to prove the document existed unmodified at that give time. This timed proof of existence is the main functionality of the so called *blockchain notary systems*. Sometimes this property is also called *proof of ownership* to stress the property of pre-image resistance of cryptographic hash functions, that implies that only someone actually owning the document could have computed the correct hash value stored on the blockchain. Moreover, if the hash value was submitted to the blockchain through a signed transaction, the signature can be used as further proof of ownership. Of course a traditional notary has to perform additional services (not including all the legally mandatory controls) like checking for parties identities and witnessing that they are the ones signing the document, confirming that the parties are aware of the content of the document and are willing to sign it, checking if a new agreement is in conflict with an existing previous commitment, etcetera. If a blockchain notary wants to completely cover a traditional notary role it needs a

more complex infrastructure built around the blockchain to provide all those functionalities (like, for example, an identity management system). Nevertheless most blockchain notary systems available today only offer the aforementioned proof of existence. Some examples of such services are shown in Table 5.1.

Name	Link	Blockchain Used
Blocksign	https://blocksign.com	Bitcoin
Bitcoin.com Notary	https://notary.bitcoin.com/	BitcoinCash
Acronis	https://www.acronis.com/en-us/business/blockchain-notary	Ethereum
Stampd	https://stampd.io/	Bitcoin, BitcoinCash, Ethereum or Dash
Stampery	https://stampery.com/	Bitcoin, Ethereum
Proof of Existence	https://proofofexistence.com/	Bitcoin
ProveBit	https://github.com/thereal1024/ProveBit	Bitcoin
Bitnotar	https://github.com/bitcoinaustria/bitnotar	Bitcoin
Bernstein	https://www.bernstein.io/	Bitcoin

Table 5.1: Blockchain notary systems.

We do note that such simple systems have some intrinsic limitations, for example the timestamp associated to the data (i.e. the timestamp of the block the data is contained in) is not accurate for two reasons. First, the block timestamp is not accurate by itself (it is the tamperable local time of the miner who builds it), and it can vary by at most two hours from the expected one. This means that a subsequent block can have a lower time than its predecessor. By tampering with the block times, a malevolent miner powerful enough may successfully invalidate the system. Furthermore, the timestamp of the block only records the time when the transaction submitting the data is accepted, not when it was submitted. This means that the two times may differ greatly and race attacks might be possible. Of course this would be an issue only in time sensitive scenarios, where the existence of a document needs to be produced and verified within minutes. Such scenarios are rare in human applications but not among machines. The time inaccuracy can be a deterrent to a possible novel application of blockchain notary systems: machine to machine proof of existence. Nevertheless, blockchain technology allows for proof of existence between automatic devices for three main reasons:

- it allows automatic recording and checks for content existence;
- it provides formal guarantees of security for the system without the need for trust between the parties;
- it greatly lowers the cost to record or check existence proofs.

The last point is especially important when coupled with automation. A traditional notary is orders of magnitude more expensive and slower than a blockchain based proof of existence system (hundred or thousands of dollars and days versus few cents and minutes). This opens the possibility for automatic devices to automatically record some information regularly in time. For example, the entire profile of a social network user could be recorded at regular snapshots in time to certify the content uploaded and activities carried on. Another field of application might be with IoT devices.

5.4.1 Intellectual Property

The real improvement over traditional systems comes when a notary system is deployed on top of a blockchain supporting smart contracts. A smart contract can not only record the proof of existence of certain data, but also define operations allowed on such data. This opens for entirely new possibilities. For example we can think of a smart notary system for the protection of intellectual property. In fact, blockchain based notary systems can prove the existence (and potentially ownership) of any digital content including music, videos, images and other user creative content (e.g. *Monegraph* that allows to record and trade digital assets on the Bitcoin blockchain [190]). Using smart contracts we could define rules for the fruition of the content, possibly coupled with a pricing of such operations (i.e. royalties) using the underlying cryptocurrency supported by the blockchain (if present). For example we could deploy a smart contract that not only provides the proof of ownership for a certain new song, but also allows users to purchase the right to listen to such song, paying directly to the owner's address. Such a system would cut all the intermediaries, allowing the owner to receive the full gain for their creations. Of course users may still have incentives to rely on third parties for advertisement, visibility, etcetera. More importantly setting up a smart contract is way cheaper than traditional means, so new users would have a very low entry cost to start offering their own creations. It also increases transparency, because the owner knows who is accessing their content, and users willing to access it have a clear and cheap entry point to ask the access permission (i.e. what is traditionally called a *license*). The company Ujo Music [265] is already offering such a service using the Ethereum blockchain. Its first proof of concept was deployed in 2015 publishing the song *Tiny Human* by Imogen Heap [142] through an Ethereum smart contract. Another similar service is *MUSE* [193] that instead of relying on an existent blockchain it builds its own.

5.5 Supply Chain Management

Problem definition. A supply chain is the chain of passages (physical or virtual relocation, transformation and exchange) that products or services undergo from raw material or natural resources (including human intellect) to a finished product for

the end customer. The management of a supply chain is a complex task that requires to plan all the activities and logistics involved and it spans across many different companies and suppliers, all the way to the customers. Often the management of a supply chain has also to take into account standards and regulations (both from states and companies) that need to be satisfied between links in the chain. In general, there is little to no trust between different nodes in the chain, since trust is built between companies as they successfully work together over time, and this gets more and more difficult in a more open and fluid market. This means that a paper trail needs to be produced and verified at each step to try to protect players from counterfeiting and keep the goods moving along the chain. This paper trail can also be required by third party inspectors to verify that no regulations have been infringed. Since the chain spans from raw material to end consumer it is often very long and stretched along different jurisdictions, rapidly becoming cumbersome and expensive to maintain, as well as prone to forgery and human error. It also may cause additional expenses to the parties involved to settle disputes or inadequate goods returns. This often means that in most scenarios the supply chain management costs have a significant impact on the final price of a traded good.

Blockchain Based Proposals. As we have seen in Section 5.4, a blockchain can be effectively used to provide a timestamped proof of existence of a digital asset to create a notary system. This same concept can be extended to a supply chain management system. By digitizing the physical goods, for example by associating them a unique tamper resistant code, and recording on the blockchain all the information associated with them (e.g. price, date, location, quality, certification, etcetera) as well as their passages between links in the supply chain, we can obtain a secure and transparent supply chain management system on top of a blockchain (see Figure 5.4). Since all information stored on the blockchain is immutable, it becomes impossible (under blockchain security assumptions) to tamper with the system. The base idea is no different from the original intuition behind Bitcoin, as in Bitcoin the blockchain is used to track the history of each coin to prevent double spending. So in a blockchain based supply chain system it is used to track the history of digital traces of assets. The main advantages of such a system are the following.

- **Increased transparency**. Storing all the immutable information on a blockchain would increase the traceability of products and so enhance trust between parties involved as well as final consumers. Clearly having an easy traceable chain of transformations of a good will help contrast counterfeits and frauds, as well as ensure intermediaries and external contractor suppliers compliance to corporate standards and regulations. The transparency of the entire process could also enhance the reputation and public image of virtuous companies.

- **Easier auditability**. Any third party having access to the traceable life of a good on the blockchain could check its correctness without need of a slow and costly inspection of an entire paper trail compiled in different formats. The end

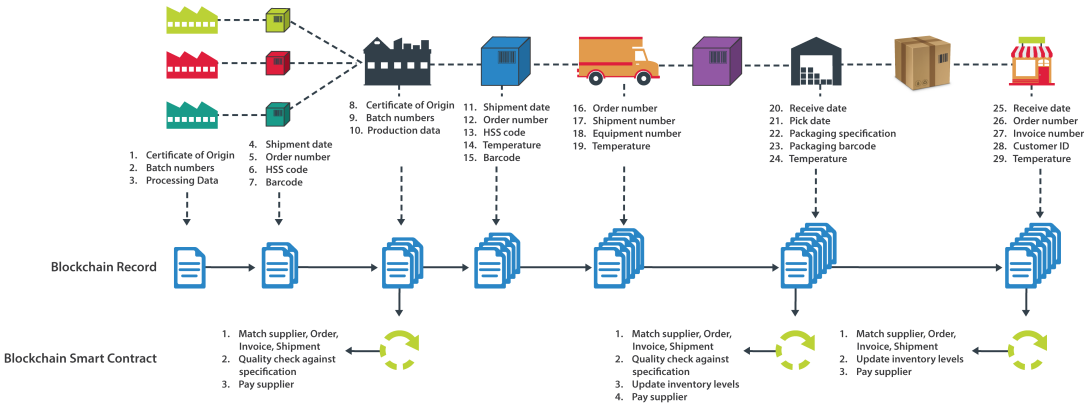


Figure 5.4: Scheme of a generic blockchain based supply chain management system. Source [39].

user itself could verify the origin of the materials used and each transformation step to buy accordingly.

- **Reduced management and verification costs**. Checking or maintaining the trace of a product on a blockchain is way less expensive than examining or creating a long paper trail, leading to huge cost reductions for the companies involved as well as independent auditors. Moreover, the security granted by the blockchain would reduce the need for many intermediaries as well as dispute resolutions, further lowering the companies expenses (and potentially the final price of a good for the consumer). Finally, the contrast to counterfeiting and grey market will further reduce honest parties losses. This is especially true for high value goods such as diamonds and pharmaceutical drugs.

- **Increased management and verification speed**. Events recorded on a blockchain can be created and audited in almost real time, speeding up both recording and verification operations. By adopting a blockchain as data recording standard cuts intermediaries and the process can be sped up as well. Furthermore, the process can be automatized through self enforcing smart contracts to speed up repetitive operations.

By lowering the initial costs and the importance of reputation for suppliers, a blockchain system could also help a more fluid and dynamic suppliers ecosystem. Paradoxically, even if a blockchain is in general more costly and inefficient than centralized solutions, in this scenario it would help to create a more competitive and agile environment.

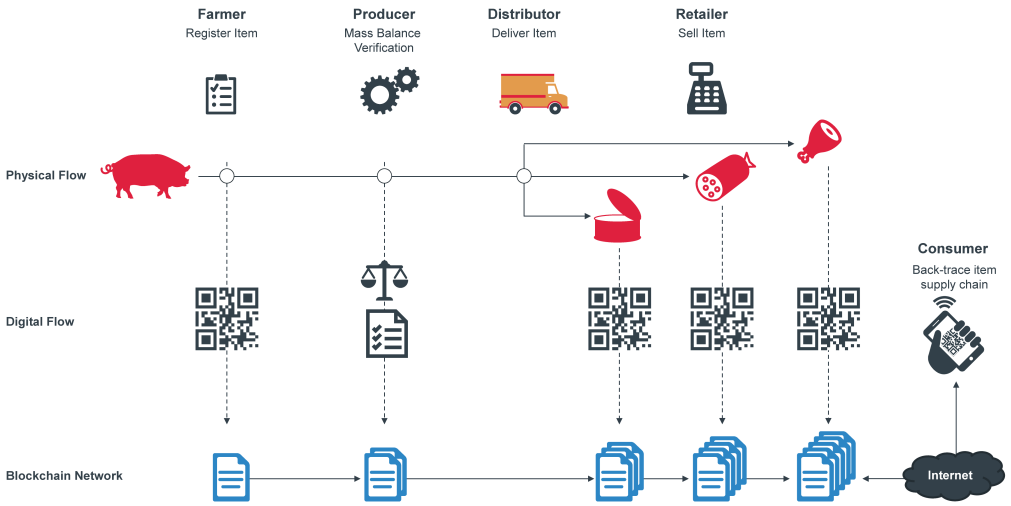


Figure 5.5: Scheme of Walmart pork tracking in China using a blockchain based supply chain management system. Source [39].

Cases Studied. In general, the property of increased and easier traceability would be beneficial for different sectors where product provenance is critical. For example, in the pharmaceutical industry it can be lives-saving to timely recognize the manufacturer of a faulty drug to fast isolate it. Another practical example is the food industry where it might be necessary to track down products to the farms or treatment plants that produced them in case of disease outbreaks or dangerous contamination or pollution. A pilot project to use blockchain technology to trace the pork industry in China was started by Walmart [63] in partnership with IBM (see Figure 5.5), leveraging the Hyperledger blockchain [52]. IBM is not new to supply chain applications of blockchain technology built on top of Hyperledger, as it is also collaborating with Maersk, leader in the ocean shipping industry, on a similar project, mainly aimed at reducing the traditional paper trail costs that Maersk estimates to be as much as one fifth of the entire shipping costs [172].

But provenance is not only important to pinpoint producers of faulty products, it is also important in industries with the need to prove that goods are authentic or ethically produced or obtained (e.g. fair trade or organic certification in the food industry). A practical application is diamond tracking. For such valuable goods it is important to prove their authenticity and that they were not used to fund violence (the so called *blood diamonds*). Systems tracking diamonds on blockchain are already used alongside the traditional methods, for example, the De Beers Group of Companies, the world largest diamonds producer by value, developed a working prototype [87]. Another older initiative is *Everledger* [105] that is active since 2015

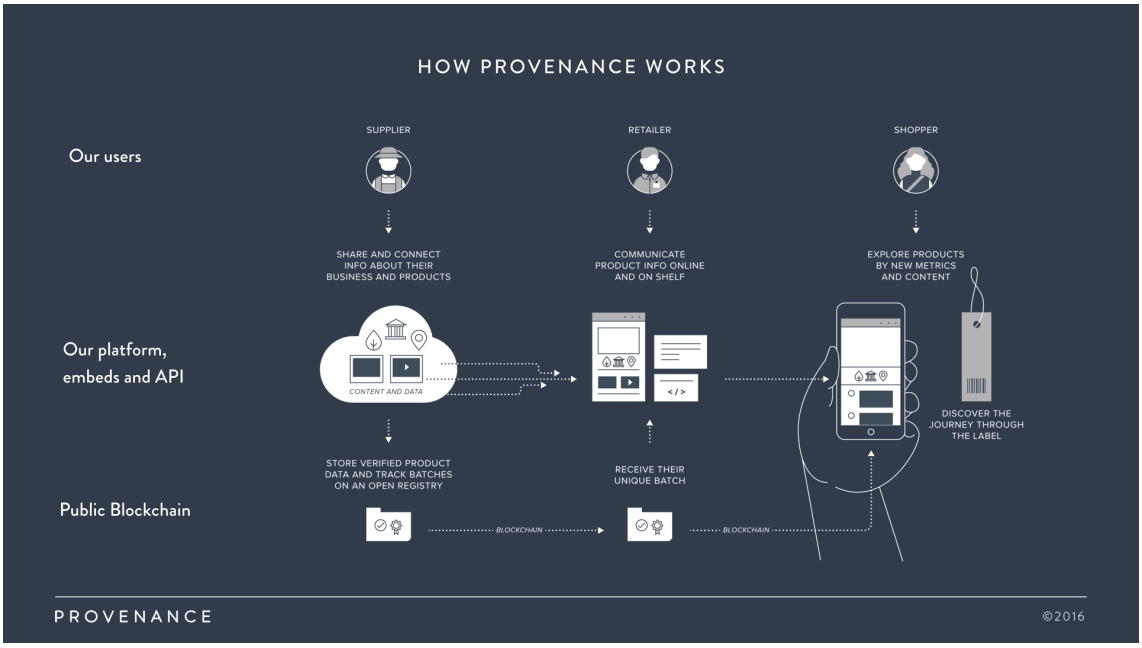


Figure 5.6: Scheme of *Provenance* blockchain based supply chain management system. Source: [220]

and allows to store arbitrary valuable goods but strongly advertises its adoption with diamonds [106]. *Provenance* [220] is another company offering general supply chain management based on the blockchain (see Figure 5.6). Its pilot experiment was the tracking of tuna in Indonesia [221] to prevent human rights abuse of fishermen as well as illegal overfishing and other examples are *Blockverify* [40], *Sweetbridge* [252] and *Factom* [110].

We should now point out that often not all information should be kept publicly visible on the blockchain. In fact, that could cause confidential information leaking that could be used by competing companies to gain an unfair advantage. But transparency is not necessarily in conflict with privacy. Advanced cryptographic techniques can be used to guarantee that some properties are satisfied by the data without revealing the data itself (e.g. zero knowledge proofs [112] and homomorphic encryption [122]). For example, a supplier might publish private information about their inventory without showing the actual amount of each product. This way a buyer could know if the supplier has a certain good available without actually knowing how many units are available, so to avoid the leaking of private corporate information that a competitor could profit from. At the same time a farmer might keep its identity private to the customer while still proving that their products satisfy a certain standard (e.g fair trade). A company providing a blockchain based supply chain management system with a keen interest in privacy is *Chronicled* [60] that relies on zk-SNARK and is focused on applications involving the IoT.

Finally, we point out how integrating blockchain supply chain systems with smart contracts could potentially bring whole new possibilities, especially if paired with IoT. A traditional or blockchain based supply chain management system can be greatly strengthened when coupled with *smart labels*, i.e. labels of goods containing IoT sensors providing a data feed elaborated by smart contracts on the blockchain. If such sensors are built tamper resistant (for example employing hardware secure chips) then they can provide the smart contracts with a secure flow of data to monitor the goods along the supply chain, detecting counterfeiting attempts or exposure to critical conditions. A clear application would be the pharmaceutical supply chain, where smart sensors could monitor that delicate drugs are kept within the intended threshold of temperature, humidity, light exposure, etcetera. An example of a company providing a blockchain based supply chain management system oriented to such smart IoT devices is *Modum* [188, 41], especially active in pharmaceutical monitoring and tracking. Of course the use of smart contracts and IoT devices in a supply chain scenario enables also real time analytics, that could be leveraged by machine learning predictive algorithms to adjust production accordingly. Monitoring in real time or predicting the goods manufactured would allow suppliers to better manage a dynamic production as well as reduce material waste and economical losses from unsold items. The benefits would not end with the end consumer buying the product, as the entire recycling and collection industry is part of the supply chain as well and would benefit from such innovations.

Even if the application of blockchain technology to supply chain management is mainly carried out by the interested companies and their relative pilot tests, also some academic work has been presented in the last two years. In [118] the authors address the inefficiency of a public blockchain supporting a supply chain management system, proposing an alternative solution. In [263] blockchain is used to contrast counterfeit and stolen product tags, by building a supply chain scheme to transfer ownership of radio frequency identification tagged products, so that each node in the chain can check integrity and correct ownership of the tags before buying the corresponding product. A proof of concept implementation of the proposal, based on the Ethereum blockchain, is presented and evaluated. [161] presents a survey of blockchain based supply chain management applications and discusses general potentialities and limitations. [157] is focused on digital supply chain, i.e. electronic data exchange between business partners focused on Business to Business integration, that is traditionally heavily reliant on third party intermediaries. Through interviews to experts (mainly to Finnish entities) the authors highlight the main perceived advantages of the applicability of blockchain technologies to such a system. In [279] a federation of private blockchains connected through a public one is presented, that allows for complete exchange of private information between interested parties (through a private chain) while leaving the process monitorable (without access to private information) through the public one. Finally, [175] is an interesting work not directly connected to the supply chain management problem but still pertinent, since it highlights a different approach to reach a common goal:

consumers information and building of trust. The authors point out the importance of reviews in current online commerce, while remembering that most review platforms are centralized entities able to malevolently forge reviews to mislead consumers. A much more secure blockchain based decentralised system to submit and read reviews is presented, and a proof of concept implementation on the Ethereum blockchain is shown and evaluated.

6

Access Control Systems

Abstract

In this chapter we present a brief background about Access Control systems. After defining what an Access Control system is and presenting its different proposed models, we give an overview of the possible benefits that a blockchain technology integration could bring. The main goal of the chapter is to present the Attribute Based Access Control model and XACML standard since those are the tools that we decided to use in our proposed systems in the rest of this thesis.

In the last years, due to the ever increasing coverage of Internet connectivity, the most part of existing digital resources (e.g., servers, data bases, services or even smart objects from smart watches to last generation cars) have been connected and exposed to, and hence accessed through, the Internet. It is obvious that, if on the one hand this connectivity enables the provision of new and better functions, on the other hand, it introduces new security risks of unauthorized accesses to the connected resources. In order to prevent privacy violations and improper, erroneous, damaging, or malicious uses of such resources, they need to be protected by proper security systems, such as Access Control Systems, which control the access attempts and grant the access only to those subjects actually holding the corresponding rights in the current access context.

An Access Control System (ACS) is the security component of an application scenario whose task is to protect the resources by checking the access requests performed by the subjects. In other words, the ACS decides whether these subjects have the rights to perform the accesses they request in the current access context or those accesses must be denied. The rights of subjects to access resources are expressed by means of access control policies, which consist of a set of conditions that are evaluated against the current access context to make the access decision each time an access request is received [237]. In some scenarios, the right of performing the access is not static, and hence it is continuously verified for the whole duration of the access itself, in order to interrupt it in case this right expires because of a change of the access context [164]. Several models have been presented in the

scientific literature to find a different way of defining the access rights, e.g., Mandatory Access Control (MAC), Discretionary Access Control (DAC), and many others. Among them, the Attribute-based Access Control (ABAC) model [269] represents the access context through a set of attributes describing the relevant features of the subjects, resources and environment, and uses access control policies consisting of a set of conditions over the values of these attributes. Examples of attributes of the subject S could be, for instance: the ID of S, the ID of the company S works for, the role of S in this company, the name of the projects assigned to S, the number of resources S is currently using, and so on. Example of attributes paired with a piece of data D could be: the project D belongs to, the privacy level assigned to such data (e.g., *public*, *internal* or *confidential*), the ID of the producer of D, and so on. A very simple example of ABAC policy could be the following: a subject S is allowed to access the document D if D belongs to one of the projects assigned to S. This policy simply compares the value of the attribute project of the subject with the value of the attribute project of the resource.

Several languages are currently available for writing access control policies, being the eXtensible Access Control Markup Language (XACML) defined by the OASIS consortium [200] one of the most popular one for expressing ABAC policies.

Resource owners can deploy and run their Access Control Systems on their own premises or they can exploit the ones provided (e.g., as services) by trusted third parties. Since the configuration, deployment and management of an Access Control System could represent a relevant cost for the resource owner, to reduce the costs, the resource owner can outsource the access control functionality to external systems. In the last years, some Access Control Systems implemented as Cloud services following the Software as a Service (SaaS) paradigm have been proposed, such as OpenPMF SCaaS [163]) and ACaaS [222]. These services often use an open-platform API, in a way such that users are not bounded to use a specific implementation, but they can exploit them to have a uniform management of policy enforcement for all the resources they own. As for the other Cloud services, each user pays the usage of the ACaaS on a per use basis. For instance, Amazon provides to the users of its Cloud platform, called Amazon Web Services (AWS)[1], an Access Control Service called Identity and Access Management (IAM)[2]. AWS IAM allows Amazon users to manage the access to their AWS services and resources by creating and managing their users and groups and by defining proper permissions to allow or deny the access to AWS resources. Moreover, AWS IAM is provided by Amazon without additional charge, i.e., the cost for using is included in the use of other AWS services by your users.

Blockchain potential advantages to Access Control Systems. As said in the previous section, the owners of digital resources, for several reasons, might be willing to outsource control of the accesses to their resources to Access Control

[1]https://aws.amazon.com/it/

[2]https://aws.amazon.com/it/iam/

Systems, provided by external and trusted third parties. In other words, instead of configuring and deploying their Access Control Systems on their own premises, resource owners simply refer to existing ones to carry out the access decision process. One of the reasons for resource owners for not hosting an Access Control System on their premises is to avoid the burden due to its configuration, deployment and management, that could represent a relevant cost for the resource owner, both in terms of hardware (the servers/VMs which run the Access Control System), software (the Access Control System product could require to pay a periodical fee), and man power (the time spent by the administrators for the Access Control System management). Adopting the blockchain based Access Control System proposed in the following chapters would relieve these users from installing and managing their own Access Control Systems, because they would exploit the blockchain to evaluate their policies and perform their decision process.

A more novel reason that could lead resource owners to the adoption of a blockchain based Access Control System with respect to one provided by a traditional third party is the transparency and auditability that are guarantee by blockchain technology. In fact, the resource owners can always browse the blockchain and control the access requests that have been performed to their resources, the values of the attributes at that times, and the resulting access decision returned as response to the requester.

At the best of our knowledge, only a few proposal of blockchain related access control systems have been presented. In [286] the authors combine blockchain and off-chain storage to build a personal data management platform focused on privacy. However the control is limited to read operations on users stored data, the work does not address the general problem of Access Control systems. [204] proposes a new framework for blockchain based access control focused on IoT. Also this proposal like the previous one, is focused on a the very specific field, i.e. IoT data protection, lacking a more general scope and applicability. A blockchain based lightweight and robust access control framework addressing the security and privacy issues in Big Data is introduced in [99].

6.1 XACML Standard

This section gives a brief description of the eXtensible Access Control Markup Language (XACML) standard and reference architecture (shown in Figure 6.1) defined by the OASIS consortium [200].

XACML is a standard defining the language written in XML to express Access Control policies (already presented in the previous section) and Access Control requests and responses. Requests are used to express the attribute values that have to be provided by the subject, in the same format as policies. For example, the subject could provide the *action-id* of the action it wants to perform as well as their *subject-id*. Responses instead contain the decision relevant to a previous request

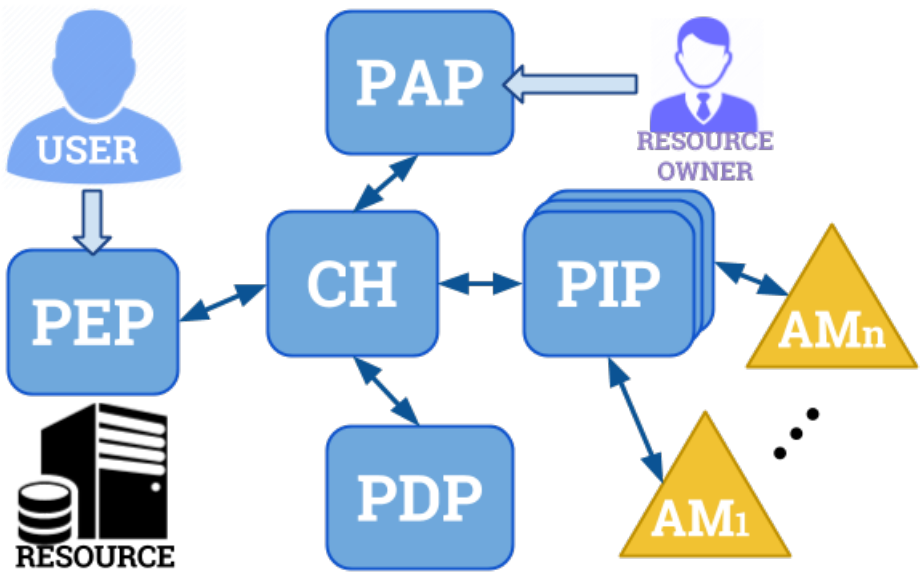


Figure 6.1: XACML reference architecture.

expressed in a fixed format. A response can be Permit, Deny, Indeterminate(in case of errors or missing values) or Not Applicable(the request does not regard any of the service policies).

XACML is based on the ABAC model (i.e. Attribute Based Access Control, see the previous section). The access decision process is based on policies and *policy sets*. A policy set is a collection of policies, other policy sets and remote policies references. Each policy instead contains a set of *rules*. Each rule expresses a *condition* that represents a boolean function over an arbitrary complex combination of (not necessary boolean) functions over a set of attributes. At request time, each rule is evaluated with the corresponding attribute values to return a decision (Permit, Deny, Indeterminate or Not Applicable). To understand which policies, policy sets or rules apply to a given request they can specify *targets*. A target is a set of simplified conditions over attribute values contained in the request.

Since policy sets may contain multiple policies each returning an access decision and policies themselves may contain multiple rules each returning a possibly different result, all these decisions need to be merged properly to obtain the final correct result. This is achieved through *combining algorithms*, either at policy set level (i.e. *policy combining algorithms*) or at policy level (i.e. *rule combining algorithms*). Each algorithm defines the way to properly merge the different individual evaluation results to produce an unique authorization decision. For example one such algorithm is the *Permit Overrides Algorithm*. It states that the final result is Permit if at least one component (either a rule or a policy) returned Permit. Few combining algorithms are available as standards in XACML (see for example Appendix C of [200]), but more can be custom defined as needed.

XACML does not only provide standards to express policies and requests/responses, it also gives a standard for the evaluation architecture [200]. The architecture scheme is shown in Figure 6.1, where is visible the interaction between the different components. In the following list we describe each component roles.

- **Policy Enforcement Point (PEP)**

  The Policy Enforcement Point is the component paired with the resource to be protected which is able to intercept and suspend the access requests, in order to perform the policy evaluation. In particular, the PEP interacts with the subjects (by collecting access requests), it triggers the decision process, and enforces the related result by actually allowing or denying the execution of the access.

- **Policy Administration Point (PAP)**

  It is the component in charge of managing access control policies. Storing and retrieving them as necessary.

- **Attribute Managers (AMs)**

  AMs are the components that manage the attributes of subjects, resources, and environment, allowing to retrieve and update their values. They could run in the authorization service itself, or they could be run in other machines in the same domain, in other administrative domains, or even by third parties. Existing services can be exploited as AMs.

- **Policy Information Points (PIPs)**

  The set of attributes required for the policy evaluation are, in the most general case, managed by a set of distinct Attribute Managers, each having its own protocol to be used for collecting the current attribute values. Policy Information Points are the interfaces for interacting with Attribute Managers, allowing to retrieve the latest values of such attributes and to update them. Hence, PIPs acts as plugins of the CH, allowing it to use a standard protocol to manage the attributes required for the evaluation of the policy, while they translates the requests to the specific protocols of the different Attribute Managers.

- **Policy Decision Point (PDP)**

  The Policy Decision Point is the evaluation engine that takes a policy, an access request, and the attribute values as input, evaluates the policy and returns the decision (i.e. Permit, Deny, Indeterminate or Not Applicable).

- **Context Handler (CH)**

The Context Handler is the component which acts as orchestrator of the decision process, interacting with the other components of the architecture to manage the workflow of the decision process.

7

Blockchain Based Policy Management

Abstract

This chapter presents our first proposal of integration of blockchain technology with an Access Control system. We propose a new approach based on blockchain technology to publish and manage the policies expressing the right to access a resource. By leveraging a cryptocurrency based blockchain we also allow the distributed transfer of such right among users. By keeping the Access Control policies publicly visible on the blockchain we provide distributed auditability, preventing a party from fraudulently denying the rights granted by an enforceable policy. We also provide a working reference implementation based on XACML policies, deployed on the Bitcoin blockchain.

We have shown in Section 6.1 how Access Control systems are used in computer security to regulate the access to critical or valuable resources, typically expressing the rights of subjects to access resources through access control policies, which are evaluated at access request time against the current access context. We have also pointed out how we will base our research on Attribute-based Access Control (ABAC) policies.

We remark now that some Access Control scenarios require access rights to be transferred from a subject to another. For instance, a user could sell its access right to another user. Another example is the one where an employee of a company who was supposed to perform a given computation on a Virtual Machine delegates the execution of this task to another employee, who needs to access that same Virtual Machine.

Moreover, the evaluation of the access control policy in order to decide whether the requested access to a resource can be executed is performed by a party which is trusted by (the owner of) that resource, but it could be not trusted for the subject of the request who, instead, would like to to be guaranteed against unduly denial of access. For example, the Access Control system could run directly on a server of the owner of the resource. In fact, the party which actually evaluates the policy and enforces the result on the resource could maliciously force the system to deny

the access to a subject although the policy would have granted it. Hence, in this scenario there is the need for the subjects to have a mean for verifying which policy has been enforced when they performed an access request which has been denied.

In this chapter we propose an approach based on blockchain technology to represent the right to access a resource and to allow the transfer of such right among users. The proposed approach is validated by a preliminary implementation based on the Bitcoin protocol.

7.1 Proposed Approach

In this chapter we propose to use blockchain technology to represent the rights to access resources and to transfer them from one user to another. In particular, we propose to store the representation of the right to access a resource in a blockchain, allowing the management of such right through blockchain transactions[1].

The main advantages of the proposed approach are:

- the right to access a resource can be easily transferred from a user to another through a blockchain transaction created by the last right owner, without the intervention of the resource owner;
- the right is initially defined by the resource owner through a transaction, and all the other transactions representing the right transfers are published on the blockchain. Hence, any user can inspect them at any time in order to check who currently holds the rights to perform a given action on a given resource. Consequently, a user who had its access request denied, can check whether the entity in charge of verifying the existence of the required right actually made the right decision.

The actors of our reference scenario are the resource owner, say P, (unique for each resource) and a set of subjects, $\{S_1, \ldots, S_i, \ldots, S_n\}$. The resource owner is the entity who has the control of the policy for each of its resources, say R_j, and it creates, updates and revokes such policies. Note that we consider for simplicity that the policy issuer is also the corresponding resource owner but, of course, our protocol works the same also in a scenario where there is a trusted intermediary entity in charge of emitting policies on behalf of the resource owner. The subjects hold the rights to perform actions on resources, as specified by the respective policies.

[1]In the following, we refer to cryptocurrency style blockchains, because they are the main application of blockchain technology currently implemented. Consequently, we assume to have the concept of transactions transferring value (expressed in the underlying cryptocurrency units) between entities with possibly some attached data. However, cryptocurrencies are just one of the possible applications of blockchain technology. Nevertheless it is always possible to define a set of tradable assets (currency) on top of any blockchain, so our approach is still usable. In those cases, we have to use the proper concept of transaction to implement the approach we propose.

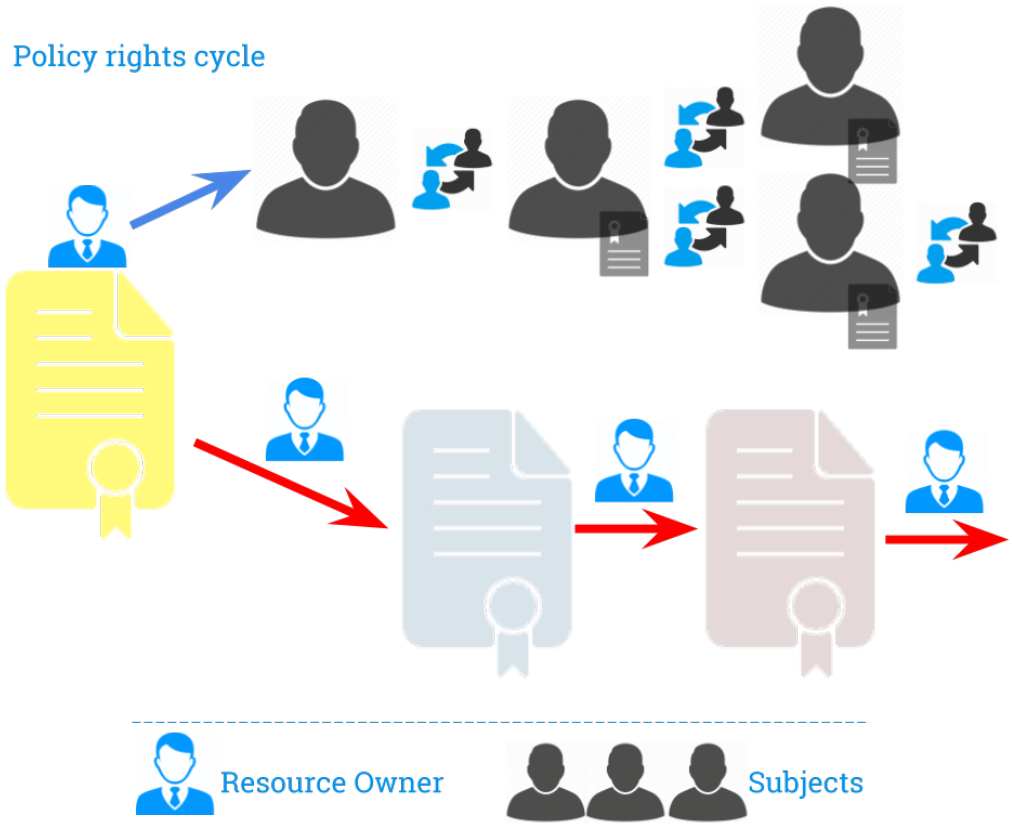


Figure 7.1: Policy rights cycle. On the top the rights exchanges between subjects, including rights splitting. On the bottom the parallel policy updates performed by the policy issuer.

The subjects can transfer the action rights specified by policies, even by refining or splitting them (as explained in Section 7.1.1).

Hence, our approach requires that P and S_i perform distinct actions, independently one from the others. The policy issuer takes no part in the policy rights exchange, and, similarly, the subject currently owning a right takes no part and needs not to be online when the policy issuer modifies the policy (even if this action might of course affect the subject right), see Figure 7.1.

7.1.1 Policy Creation, Update, and Revoke

The ABAC policy which defines the access rights on the resource R is defined by the resource owner P, and it is stored in the blockchain through a new transaction called Policy Creation Transaction (PCT). After its creation, a policy can be updated by P through a transaction called Policy Update Transaction (PUT) any number of times and, at the end, it can be revoked, i.e., canceled.

In our approach, the policy consists of two sets of conditions:

- the condition which defines the ID of the subject to whom the policy grants the access right;
- the conditions which define the sets of values allowed for the attributes of the subject, resource and environment for the access to be granted.

In other words, the resource owner decides the subject to whom it wants to initially grant the access right and a set of conditions that must hold to grant the access. In our scheme we allow these conditions to be properly modified by the right holders when they transfer these rights to other users. By *properly modify* we mean that the current right holder is allowed to:

- add new conditions in AND with the conditions already defined in the policy;
- split the set of values allowed for an attribute by an existing condition C of the policy in two (or more) sets by defining proper disjunct conditions, C_i and C_j. i.e. the set of attribute values which satisfy C_i OR C_j is the same set of values which satisfy C, and there is no value of the condition attribute which satisfies both C_i and C_j.

We note that adding conditions (done by a right holder) is not the same as executing policy updates (doable only by the policy issuer). Since the conditions added to the policy by right holders (not known in advance at policy creation time) are combined with the existing ones through an AND operator, the resulting overall policy can only be more restrictive than the original one. This means that the original policy conditions cannot be violated. During a policy update step, instead, the meaning of the policy can be completely changed. This is correct since the policy issuer is the only one that can update a policy. We also remark as the conditions added by a right holder are added incrementally for each exchange of rights, so they cannot be modified by the new right owners. This is correct since a right owner should be allowed only to restrict the rights it wants to transfer, not to expand them.

For the sake of simplicity, we suppose that each policy concerns one subject ID only. This is not a limitation, because when P wants to grant the access to a resource to several subjects, it can simply produce a distinct policy for each one of these subjects. Moreover, in our approach we suppose that each policy includes one rule only, and this rule includes all the conditions of the policy, properly combined with AND and OR logic operators.

We remember that a blockchain can be seen as a distributed append-only database replicated among all the users.This means that every piece of data added to the blockchain cannot be subsequently removed and it will constitute a permanent burden on the entire network. This is the reason why, when defining a new protocol, we should try to minimize the amount of data saved on the blockchain, storing essential information only. The problem with our approach is that the standard policy

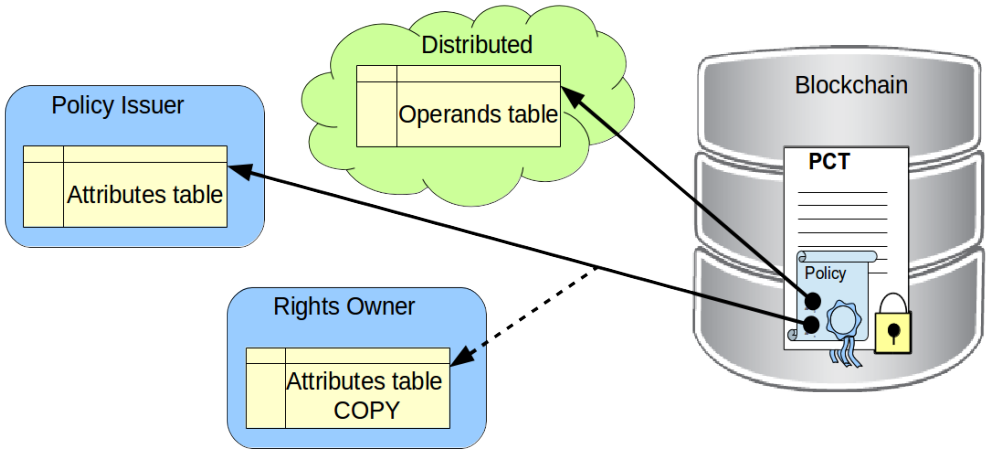


Figure 7.2: Proposed hybrid policy storage approach.

language XACML is a very verbose formalism and policies can be relatively big. Storing policies in XACML format directly on a blockchain will result in a serious storage space problem.

The easiest solution would be to store in the blockchain only a link to an external source containing the policy, coupled with a cryptographic hash of the policy itself to make it tamper proof. For example, the blockchain could save only a tinyurl or a torrent descriptor pointing to an external source hosting the actual policy (written in a standard format as, for example, XACML) [287]. The advantage of this solution is obviously to minimize the quantity of information to be stored on the blockchain, since the space occupation of the policy is constant independently of the policy size. The main disadvantage is that policies themselves are stored outside of the blockchain, thus not benefiting of blockchain technology advantages (i.e. availability, security, etc.).

Our approach (shown in Figure 7.2) adopts an hybrid solution between saving in the blockchain the entire policy or just a link to it. We chose to store policies directly in the blockchain but coded in a custom built efficient format that favors compressing policies representation and avoids information repetitions.

First we rewrite a policy expressed directly in ABAC format as a list of basic conditions over attributes. Each condition can be written as three pieces of information:

- the right attribute name;
- the operand connecting right and left term;
- the left term that can be either an attribute name or a constant value (possibly a set of constant values).

Conditions are combined through the logic operator AND/OR to form a unique condition.

If we want the policy storage to be scalable in the size of the policy we would want each of the above listed information of a condition to be represented with constant size. The logic connector of the policy is of course easy to codify with one bit (0 for OR and 1 for AND). To try to compress the rest of the condition as much as possible, we want to compress both attribute names and operands in a fixed size field (for example one byte). To compress operands we can define a protocol defined table of symbols representing the mapping between every possible operand usable in a policy and a numerical code. This map would be maintained at protocol level (open source) and updated with new usable symbols during future protocol versions. We can then follow a similar approach to map attribute names to a short numerical value. The difference is that attribute names are different between users and so the mapping have to be defined by the policy issuer. The attribute mapping is a publicly available mapping of attribute names (identifier in the verbose XACML format) with one unique code of fixed size (for example one byte). The list to be validly published (and accepted by other users to be used in policies) must be signed by the issuer. This public key/identity should be the same used to create new policies using such mapping in the blockchain. A cryptographic hash of this list is then inserted in every policy using the attributes of the list. Such hash is necessary to know what mapping is being used and it prevents the policy issuer from creating a new mapping, potentially changing the meaning of an already existing policy. The policy issuer could still delete the mapping at a future point (since it is stored locally and not on the blockchain), so it is recommended for the user buying the rights derived from a policy to locally save the corresponding mapping. In case of future dispute the right owner can prove that the mapping is correct because the hash matches and the policy issuer cannot deny to be the mapping creator because of the signature attached to the mapping.

We note that this solution allows to save verbose information about an attribute off the chain, so without space constraints. For example we can save the attribute values type, making the type of operand non ambiguous (i.e. for example differentiating an equality over integers from an equality over strings).

If we adopt this solution, the left term of the condition is the only one of potentially variable length. If it is a parameter name it can be represented as a reference to an entry of the issuer attribute table as for the right term, but if it is a constant value we need to represent it directly, eventually in a compressed format. Furthermore, since we know the type of the attribute (expressed in the verbose attributes table) we can save the values in a suitable format. For example we would save a number or a date in a numerical representation rather than in its string representation.

7.1.2 Right Transfer

A relevant feature of our approach is that the right to access a resource R can be transferred from the subject who is the current right holder, say S_i, to another subject, say S_j, through a custom data structure stored in in the blockchain, called Right Transfer Transaction (RTT). Each RTT must contain a (direct or indirect) link to the policy whose rights are being exchanged. It is worth noting that the only parties involved in a RTT are S_i and S_j, the RTT is created by S_i, and so the intervention of the owner of the resource is not required during any rights transfer.

When transferring its right through a RTT, S_i can modify the mutable conditions regulating its right only by restricting them. For instance, supposing that a changeable condition defined by the resource owner (or by the previous right owner) states the access can be performed from 9.00 AM to 5.00 PM, S_i could transfer this right to S_j by restricting the access time from 9.00 AM to 1.00 PM. S_i can also split its right in two (or more) parts, and transfer a part of it to a subject, and the other part to another subject. With reference to the previous example, S_i could transfer the access right from 1.00 PM to 5.00 PM to a third subject S_h.

We note that the subjects are only owners of rights to perform actions, in general they have no other right neither on the policy nor on the resource. We also remark that the subjects are able to freely exchange action rights between themselves without any interaction with the policy issuer. That implies that the policy issuer (in general corresponding to the resource owner) has no knowledge in advance of which subjects will be the policy right beneficiaries (even if it can of course model a subject prototype by specifying the correct attributes conditions to be satisfied inside the policy).

We also note that policy updates from a resource owner can potentially change the meaning of a policy. This means that subjects can gain rights on a certain resource that can be later changed by the policy issuer, but, since the blockchain never forgets and timestamps both the right transfer and the policy updates, those changes are manifest and traceable.

7.2 Architecture of the Proposed Framework

The architecture of the framework we propose for the enforcement of blockchain based access control, shown in Figure 7.3, is based on the XACML reference architecture (see Section 6.1), which has been integrated with blockchain technology. Specifically, in order to allow the enforcement of blockchain based access control policies, we customized the Policy Enforcement Point (PEP) and the Policy Administration Point (PAP). The resulting workflow is hence an extension of the standard one.

When requesting to perform an action on the resource, beside the IDs of the subject, of the resource, and of the action, the PEP must also retrieve an additional

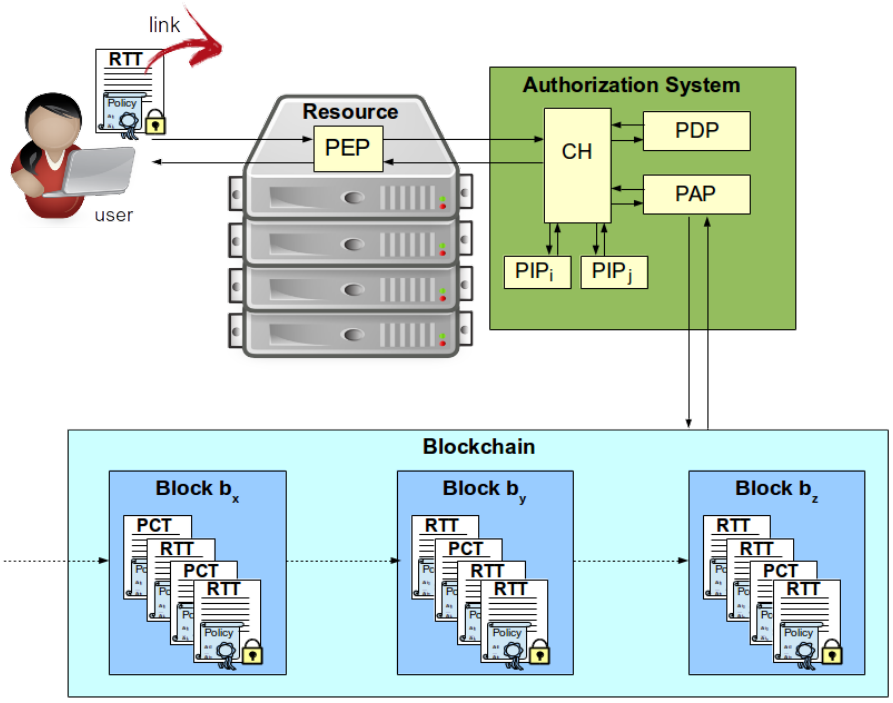


Figure 7.3: Architecture of the Blockchain based access control framework.

information to unequivocally link the subject S_i with a RTT in the blockchain. As an example S_i might be required to sign a challenge nonce with the private key corresponding to the identity it used to get the access rights in the RTT. This is no different from a classical authentication scheme in a classical access control scenario. All those information are properly included in the request which is passed to the Context Handler (CH). The CH is in charge of managing the workflow of the decision process, interacting with all the other components of the authorization system.

First of all, the CH sends the request to the PAP. The PAP extracts the RTT link from the request, and retrieves from the blockchain this RTT and all the other RTT related to this policy, as well as the initial policy and the related policy updates issued by the resource owner. The PAP combines the retrieved data to produce a standard XACML policy, and sends this policy back to the CH.

Once the security policy has been reconstructed from the blockchain and verified, its evaluation against the access request follows the process defined by the XACML standard and described in [200]. Briefly, the CH asks the Policy Information Points to retrieve the relevant attributes, it embeds these attributes in the original request, and it passes the policy and the new request to the Policy Decision Point (PDP), which evaluates it and returns to the CH the decision: permit or deny. The CH then forwards the decision to the PEP, which enforces it on the resource by executing the

request or not.

7.3 Bitcoin-based Implementation

This section describes an example of how the proposed model is deployable in a blockchain technology model. In particular, we developed a proof of concept implementation scheme based on the Bitcoin blockchain. Aim of this section is also to show how our protocol can be immediately deployed on top of an already existent real world blockchain, as the Bitcoin blockchain is, without any modification to the underlying blockchain implementation required. As a proof of the feasibility of our proposal, we report in Figure 7.4 a real PCT Bitcoin transaction we broadcasted in the Bitcoin network as publicly visible from the site https://blockexplorer.com.

In our scenario firstly a resource owner creates a new policy. Then, an arbitrary number of policy updates and right transfers can be executed, where each of the two actions can be performed independently of the other one. Finally the resource owner can revoke the policy. In our implementation each step (policy creation, policy update, policy revoke or right transfer) is performed atomically by a single Bitcoin transaction.

7.3.1 Storing data

As described in Section 1.2, the Bitcoin blockchain was designed to be used as a distributed ledger to manage a very specific kind of data: value exchange transactions. In other words, such blockchain was not designed to store arbitrary data. To overcome this limitation, we employ two commonly used methods based on Bitcoin transactions scripting language (see Section 1.2.2) to store arbitrary data on the blockchain: the `OP_RETURN` script op code and the `MULTISIG` transactions (either through a `MULTISIG` output script or a multisignature `P2SH` output) [134].

Storing data exploiting the `OP_RETURN` op code is straightforward, since it only means to insert a `OP_RETURN` code at the end of the output script of a Bitcoin transaction, followed by 80 arbitrary bytes. Storing data with a `MULTISIG` transaction is instead more complex. A `MULTISIG` transaction is a transaction with a multisignature $m \rightarrow n$, which means that it provides a list of n addresses that can sign the transaction (to spend it) and a redeem transaction will be deemed valid if at least m of these addresses sign it. To use this kind of transaction to store data we use $1 \rightarrow n$ multisignature transactions (so any one of the addresses specified can alone redeem the transaction) and in the list of n addresses we insert only the first address as a valid address and use the remaining $n - 1$ address fields to store our data. This works because the first address is enough to redeem the transaction. By the Bitcoin protocol specifications each address takes 32 or 20 bytes to be stored and each `MULTISIG` transaction can have at most fifteen addresses to be considered standard, allowing us to store up to 480 bytes of data in each `MULTISIG` transaction

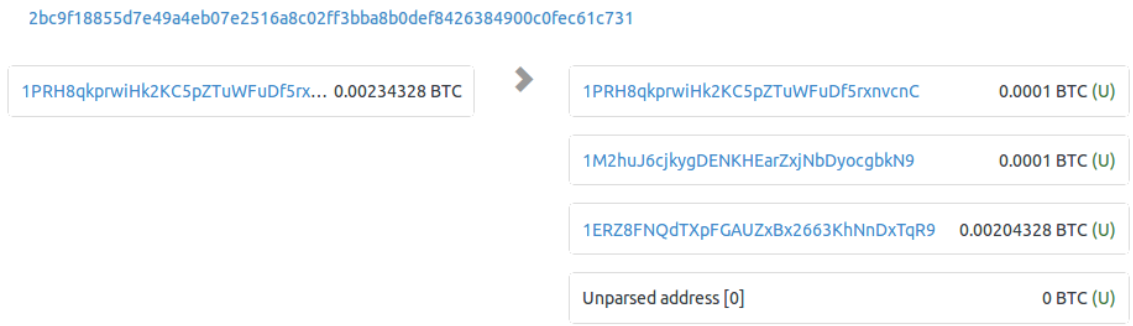


Figure 7.4: A real example of PCT in our Bitcoin-based proof of concept implementation.

output. Note that even if the data is stored in 32 (or 20) byte sized separate chunks we simply concatenate all the chunks together to obtain a single uninterrupted data stream.

Without going in further details we only note that our implementation automatically chooses the method to be exploited without the need of user intervention. In particular, in case of a write operation (i.e., the creation of a new Bitcoin transaction) our approach chooses the correct option based on the size of the data, while in case of a read operation (decoding of a transaction created by someone else) our approach first scans the transaction script type to find which of the two methods has been used to produce the transaction and then it reads the data accordingly. Whatever storage method we use the policies and conditions data is encoded in a compressed custom format that follows the hybrid approach showed in Section 7.1.1.

Since each step is performed exploiting a Bitcoin transaction, each step has a price, i.e., the price of the underlying transaction, defined as the voluntary transaction fee paid by the transaction, which is dependent on the transaction size [28]. So we can evaluate the cost of a step as the size of the underlying transaction necessary to perform it. As noted in Section 7.1.1, the size of the transaction does not only determine a cost for the transaction creator (determining a proportional fee), but for the entire network. In fact, once accepted in a definitive block, a transaction will be remembered forever in the blockchain and so stored by every user (running a Bitcoin full node). Every Bitcoin full node also keeps in its main memory a data structure to keep track of all unspent transactions outputs (UTXO), so if we include a big output in a transaction (e.g. by including a big multisignature output) this will also encumber precious main memory space of all the users. So our aim during the implementation has been to minimize the extra data saved in the blockchain, not only to keep the fees cost low, but also to encumber the network as little as possible. Finally, we point out that during each step the transaction price is payed by the beneficiary of the action. For policy creation, revoke and update transactions the price is payed by the policy owner (that is the one benefiting from such operations), while for a rights exchange the price is payed by the buyer (since the buyer is the

one who will benefit from the rights).

To embed data in a transaction we first need to create a Bitcoin transaction and so we need value to be exchanged. To build transactions we will use fixed amounts of BTC to represent *tokens*, using an approach similar to the ColoredCoins proposal [29]. We call them tokens because the value they represent will be used in transactions to carry data through the connected scripts, so we are not interested in the monetary value they represent but rather on the information they carry (visible only to those who take part in our protocol). The actual trade value of such tokens is completely independent from their nominal value (i.e. the number of BTC they represent). The fixed amount chosen for a token should be low enough so that it is easy to be owned by any user (otherwise only rich users could take part in the protocol) and its economical value is not relevant compared to its protocol specific value, but also high enough so that it can be transacted freely between users (above the dust limit [68]). In our current implementation we chose 0.0001BTC that corresponds to few euro cents at the exchange rate at the time of developing this work. We will indicate this value as *CommonAmount* in the rest of this chapter.

7.3.2 Policies Management

Policy Creation

A new policy is issued by the resource owner by creating a new Bitcoin transaction with one or more inputs and two or three outputs (see Figure 7.5). Each of the first two outputs will create a new token, so it is paying out the value of CommonAmount. The only purpose of the inputs is to provide enough funds to create these two tokens and so should include any number of resource owner funds so that $\sum(input\ values) \geq 2 * CommonAmount + fee$. The first two outputs are mandatory, and their structure is defined by the protocol, while the third output is optional, and it represents the change address for the resource owner to keep the unspent input (i.e. $\sum(input\ values) - 2 * CommonAmount + fee$) . The order of the first two outputs is important (it can not be changed):

- the first output creates the token that will be subsequently used to perform rights exchanges among subjects. It is credited either to an address that will be used by the policy issuer to sell the action rights to the first subject, or to the first subject directly.

- the second output creates a token containing as data the policy encoded in our custom format. This token is credited to an address controlled by the resource owner and it will be used by the policy issuer to update/revoke this policy in the future.

When the resource owner creates this transaction, the network is notified and, eventually, this PCT will be inserted in the blockchain. If the policy is too large

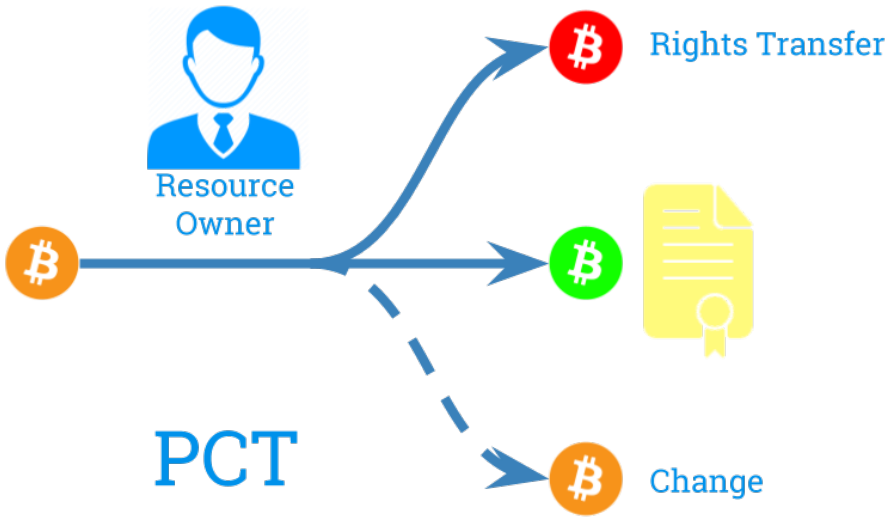


Figure 7.5: Scheme of a PCT transaction.

to be included in the second output data field, the policy issuer creates a normal PCT and then creates a chain of policy update transactions (as explained later) to include all the information required. We note that the policy creator does not have to wait for the PCT to be included in a block before starting to create policy update transactions, since they are the owner of all input and output addresses in both policy creation and update transactions and, consequently, there is no risk of double spending attempts. In the end, this means that a very long policy will generate several transactions and, consequently, it will be simply more expensive for the owner (due to more fees to pay).

Policy Update/Revoke

At any time the policy issuer can update or revoke a policy it created before. To do so, it creates a new transaction (named PUT) spending the second output of the creation policy transaction if the policy was never modified before, or spending the output of the last update policy transaction if the policy have been already updated at least once. Obviously, only the policy issuer can create those transactions because it is the only one that can spend the corresponding output.

- *Update:* the PUT transaction has two (or more inputs). The first input corresponds to the previous PUT or PCT output and the additional inputs are meant to provide the value necessary to be spent as fees to pay for this transaction. The transaction has one or two outputs, the first one carries on the token of the previous policy update or creation step, while the second one is only used as change address to collect the money left after paying the transaction fees. The update token contained in the first output is used to store the data containing the policy update information (see Figure 7.6).

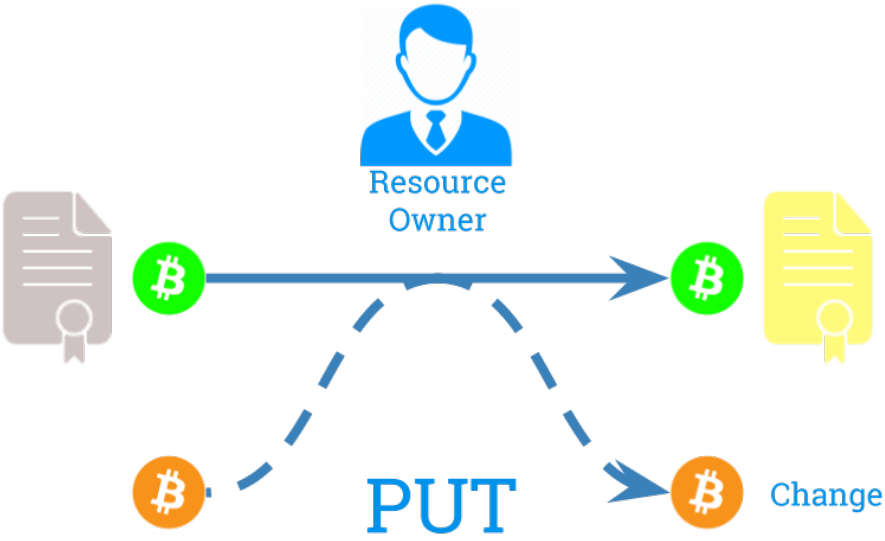


Figure 7.6: Scheme of a PUT transaction.

- *Revoke:* to revoke a policy, the policy issuer must spend the related token (even to himself), i.e., it must use it as value instead of using the embedded information. To this aim, it just creates a transaction spending the input corresponding to the previous PUT or PCT. This effectively destroys the token, thus canceling the policy.

7.3.3 Rights Exchange

To allow the exchange of access rights between two (or more) subjects we assume the existence of some kind of marketplace (or any way of exchanging messages between users) where subjects interested in selling or buying action rights take part. We also note that, since each policy and its updates are publicly visible in the blockchain, each subject can first check a policy to verify the actual rights it is buying. The right exchange between two subjects is achieved through the participation of the subjects in a message exchange protocol to allow them to jointly build and sign the RTT. Main goal of the message exchange is to guarantee that both subjects sign the RTT only after checking that it fulfills the exchange agreement. The RTT is basically a transaction where the token representing the access right is passed from the current subject to the new one and, in exchange, the new subject accredits some money (expressed in BTC) to the current owner (see Figure 7.7). Furthermore the token can be enriched by the old owner with new data to refine the policy conditions and it can be divided in different tokens (as explained in Section 7.1.1). We have seen in Section 7.3.2 that the right transfer token is created initially by the resource owner in the policy creation transaction, this means that the resource owner is the first one to sell the rights to a subject.

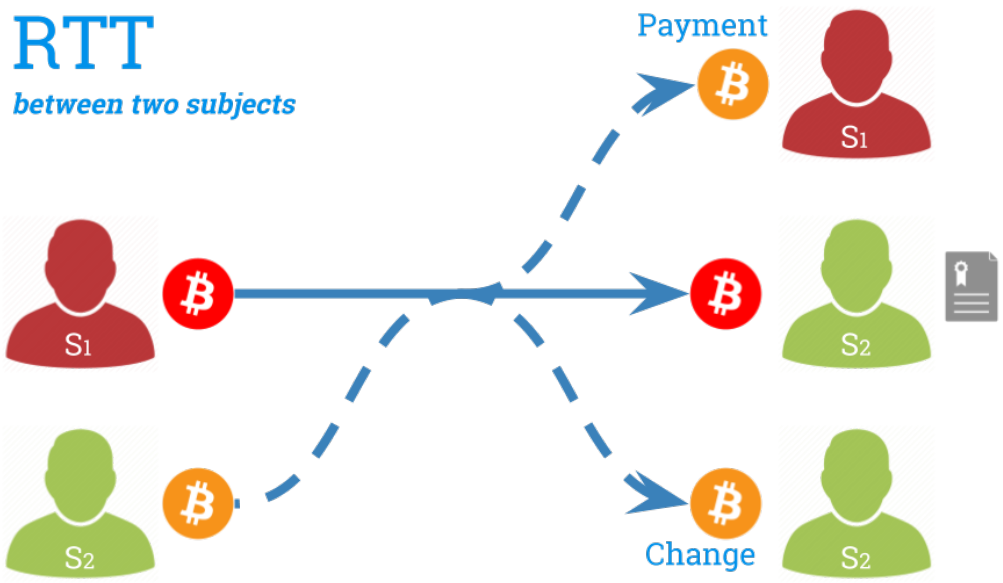


Figure 7.7: Scheme of a RTT transaction between two users and no rights restriction.

Note that the fact that rights are represented by a token, coupled with the fact that every output can be spent only once, guarantees that the same rights can be transferred only once. Note also that the subject that has currently the policy action rights can also decide to destroy those rights. To do so it only needs to spend the corresponding token as if it was just normal value (using the same process explained previously used by the policy owner to revoke a policy). This is semantically correct since the current owner has payed for the rights and so it can do with them whatever it wants. It could as well decide to never sell the rights again, which is the same for the other users as if it had destroyed them. The advantage is that the resource owner can see from the blockchain when a subject rights token has been destroyed, and so it could choose to revoke the old policy and issue a new one. We also note that revoking a policy or destroying subject rights actively removes the policy data heavy outputs from the UTXO (see Subsection 7.3.1) of all the users, so any policy stops encumbering the network once it is not active anymore.

7.3.4 Policy Evaluation

Let us suppose that a policy granting access rights to the resource R has been created and updated m times, and that this right has been transferred among subjects n times. This means that the blockchain includes a PCT, say pt, defined as in Section 7.3.2 with a chain (actually a tree in case of rights splits) of n RTT defined as in Section 7.3.3 originating from the first input of pt and a single chain of m policy update transactions originated from the second output of pt (see Figure 7.8).

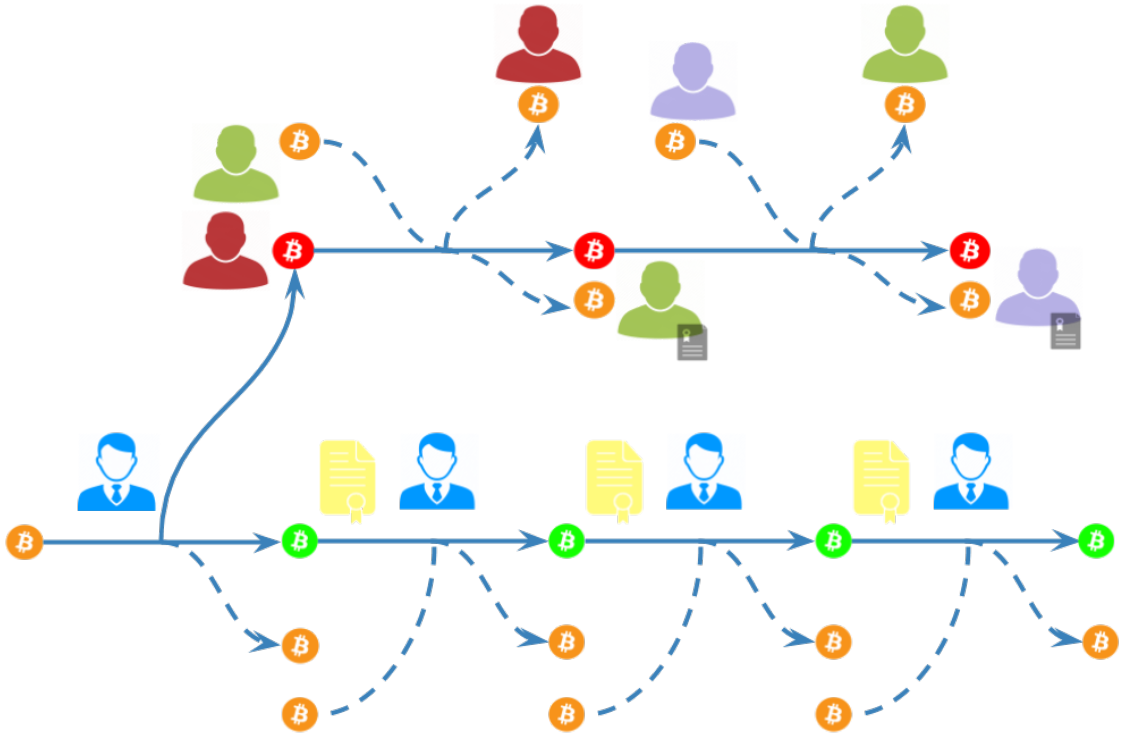


Figure 7.8: Scheme of the transactions during a policy life cycle. It is started by a PCT, then a chain of right transfers through RTT and policy updates through PUT can coexist independently from each other.

When the PEP receives a request, it only receives a link (for example a cryptographic hash) to the last RTT, say rt, and this is the only information it needs to pass forward in a request to the CH (see Section 7.2). Given a request the PAP can access the blockchain and navigate backward the chain of n RTT from rt all the way back to pt, collecting at each step the additional conditions added by right owners. Once the PAP has reached pt it can read the policy from the blockchain. Then it traverses forward the chain of all m policy update transactions, updating the policy accordingly with the data read at each update step. Once it has the fully updated policy it can add the restricting conditions inserted by right owners and read during the RTT chain traversal (see Figure 7.9). At the end of this process the PAP has derived the completely updated policy in a standard format ready for evaluation by the PDP.

Note that the above policy reconstruction can be done by anyone, given a RTT, since all the information are publicly visible in the blockchain. This is particularly important for the interested subjects that can retrieve the same way the updated policy from the blockchain and then decide whether to buy the rights for themselves or not.

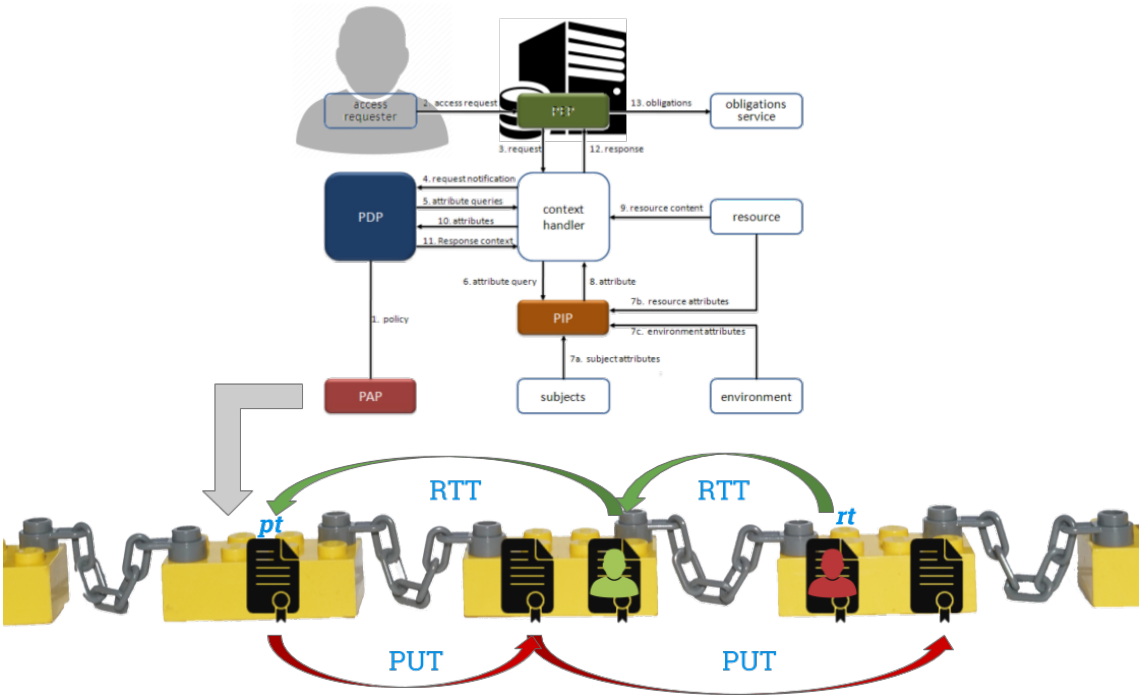


Figure 7.9: Scheme of the evaluation architecture showing the policy retrieval backward from the last RTT rt and forward through all PUT.

8

Blockchain Based Access Control Services

Abstract

This chapter presents a new design approach for access control services leveraging smart contracts provided by blockchain technology. The key idea of our proposal is to codify Access Control policies as executable smart contracts on a blockchain, hence transforming the policy evaluation process into completely distributed smart contract executions. Not only the policies, but also the Attribute Managers required for their evaluation are smart contracts deployed on the blockchain. We present two different reference examples depending on the type of resource we plan to protect. In fact, also the resource itself can be a smart contract, leading to a fully integrated blockchain Access Control system. To prove our approach feasibility we introduce our reference implementation using XACML policies and Solidity written smart contracts deployed on Ethereum. Finally, we evaluate the advantages and drawbacks of the proposal, evaluating the system performances through experimental results of our reference implementation.

In this chapter we present our design, implementation and validation of an innovative attribute based access control system built on top of blockchain technology. The proposed system follows the reference architecture defined by the XACML standard (see Section 6.1), and its innovative feature is that most of the XACML reference architecture components are implemented as smart contracts which are deployed, stored and executed on a (smart contract based) blockchain. For instance, the access control policy is represented as a smart contract which is deployed on the blockchain through a proper transaction and it is executed every time an access attempt is performed by issuing transactions, that activate the policy smart contract in order to make the access decision. Blockchain technology is also exploited for the implementation of Attribute Managers, i.e., for the management of the attributes required for the evaluation of the access control policy. The full details of the proposed system are given in Section 8.2.

With respect to the previous chapter, where we gave a description of a simple idea to integrate blockchain technology with Access Control, this chapter presents a

number of novelties. The system presented in the previous chapter leveraged blockchain technology only to strengthen the policy management process, but had no influence on the policy evaluation one (except for recovering the policy from the blockchain). In this chapter instead we will show a fully blockchain integrated Access Control system. First of all we give an exhaustive description of the system architecture, compared with the XACML reference one, and of our reference implementation. We also manage a new reference example concerning the controlling of the access to (other) smart contracts deployed on the blockchain. Since we need to use smart contracts we present an Ethereum based proof of concept implementation (instead of Bitcoin as in the previous chapter) and show a set of experiments conducted on Ethereum based testnets to validate the proposed system and to evaluate its performances.

The proposed approach presents some relevant advantages with respect to traditional Access Control Services. For instance, the subjects issuing access requests are guaranteed against unduly denial of access. In fact, adopting a traditional access control system, the party which actually evaluates the policy and enforces the result on the resource could maliciously force the system to deny the access to a subject although the policy would have granted it. Instead, in our blockchain scenario, the subjects are enabled to verify how the policy has been enforced when they performed an access request which has been suspiciously denied. We will better highlight both advantages and drawbacks in Section 8.6.

This chapter is structured as follows. We first present in Section 8.1 two reference examples that we use as example applications for our proposal. In fact we present in Section 8.2 the main ideas behind our proposal to implement an Access Control service integrated with blockchain technology. In that section we introduce the general concepts as well as the common components, revolving around the definition of smart policy, independent from the application scenario. In fact, the base components of the service are independent from the scenario they are used in, but other can not be. For example, the type of resource protected deeply influences the nature of some modules (inevitably the PEP that needs to interact with it). This is why we present the two reference examples with different types of resources, one with traditional resources, and the other with blockchain based resources, such as smart contracts. In the following Section 8.3, we present the tools we will use for our reference implementations in the rest of the chapter, including the SMART POLICY translation module implementation. We then explain in detail how the two reference examples introduced in Section 8.1 can be modelled and implemented with our proposed model. In particular, we show the application to our first example scenario in Section 8.4 and second one in Section 8.5, for both we also present our working reference implementation. We provide some considerations about our proposal in Section 8.6 and present our experimental results in Section 8.7.

In this chapter are used many terms and acronyms introduced in the previous chapters or newly defined, so, to increase the chapter readability we provide a recap of them for readers reference in Table 8.1.

TERM	MEANING
PEP	Policy Enforcement Point, see Section 6.1.
PDP	Policy Decision Point, see Section 6.1.
PIP	Policy Information Point, see Section 6.1.
PAP	Policy Administration Point, see Section 6.1.
PTP	Policy Translation Point, module of the PAP in charge of translating an XACML policy into an executable SMART POLICY.
AM	Attribute Manager, see Section 6.1.
CH_E	Module of the Context Handler, see Section 6.1, in charge of starting the evaluation process to evaluate an access request.
CH_D	Module of the Context Handler, see Section 6.1, in charge of deploying a new smart policy into the underlying blockchain.
CH_B	Module of the Context Handler, see Section 6.1, in charge of coordinating the SMART POLICY execution.
SPT	SMART POLICIES Table, used to remember the pairing between SMART POLICY addresses and resource IDs.
PCT	POLICY CREATION TRANSACTION, transaction to deploy a new SMART POLICY.
PUT	POLICY UPDATE TRANSACTION, transaction to update an existing SMART POLICY.
PET	POLICY EVALUATION TRANSACTION, transaction to trigger a SMART POLICY evaluation.
SMART POLICY	Access Control policy expressed into an executable format as smart contract.
SMART AM	Attribute Manger implemented through a smart contract.
SMART RESOURCE	Smart Contract on whose access needs to be controlled.
MATCHE	<Match>..</Match> element of an XACML policy, used as finest granularity to estimate policy complexity.
LINKER	Fixed contract used to link a SMART RESOURCE with a SMART POLICY.
evaluate	Main function of a SMART POLICY contract, used to obtain the policy evaluation result.
RIP	module in charge of inlining the PEP code into the SMART RESOURCE.

Table 8.1: Main acronyms and terms used and introduced in this chapter.

8.1 Reference Examples

This section presents two relevant still different reference examples of application scenarios for the blockchain based Access Control System proposed in this chapter. These examples will be used through the chapter in order to help the reader understand the proposed system. The first reference application scenario concerns the control of accesses to traditional digital resources. In particular, we take into account the accesses to a streaming video service for consuming digital contents. Do note that by *video streaming service* we intend it as video content sharing broadcasting and not live video streaming. We refer to this reference application scenario also as *hybrid scenario* in the following (to stress the fact that is deployed part on the blockchain and part off-chain).

A possible access control policy for this scenario could state that users having the role "premium customer" are allowed to open contents that can be chosen among the standard catalogue or latest entries, without limitations on the number of contents that can be opened. Instead, users having the role "basic customer" are allowed to open contents from the standard catalogue without limitations, while they can open a content belonging to the latest entry catalogue only if that content is not being

accessed by other basic customers at request time.

An example of an XACML policy for the streaming video service scenario is shown in Listing 8.1.

```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" PolicyId="
    videoStreamingServicePolicy" RuleCombiningAlgId="urn:oasis:names:tc:xacml
    :1.0:rule-combining-algorithm:first-applicable" Version="1.0">
  <Description>regulates the accesses to a video streaming service</Description>
  <Target></Target>
   <Rule Effect="Permit" RuleId="premium-customer-rule">
     <Target>
        <AnyOf>
           <AllOf>
              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                  >
                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                     ">premium</AttributeValue>
                 <AttributeDesignator AttributeId="subject:customer-role" Category
                     ="urn:oasis:names:tc:xacml:3.0:attribute-category:subject"
                     DataType="http://www.w3.org/2001/XMLSchema#string"
                     MustBePresent="true"></AttributeDesignator>
              </Match>
              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                  >
                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                     ">open</AttributeValue>
                 <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:
                     action:action-id" Category="urn:oasis:names:tc:xacml:3.0:
                     attribute-category:action" DataType="http://www.w3.org/2001/
                     XMLSchema#string" MustBePresent="true"></AttributeDesignator>
              </Match>
           </AllOf>
        </AnyOf>
     </Target>
  </Rule>
  <Rule Effect="Permit" RuleId="basic-customer-rule-standard">
     <Target>
        <AnyOf>
           <AllOf>
             <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                     ">basic</AttributeValue>
                 <AttributeDesignator AttributeId="subject:customer-role" Category
                     ="urn:oasis:names:tc:xacml:3.0:attribute-category:subject"
                     DataType="http://www.w3.org/2001/XMLSchema#string"
                     MustBePresent="true"></AttributeDesignator>
              </Match>
              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                  >
                 <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                     ">standard</AttributeValue>
                 <AttributeDesignator AttributeId="resource:resource-catalogue"
```

```
                    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:
                    resource" DataType="http://www.w3.org/2001/XMLSchema#string"
                    MustBePresent="true"></AttributeDesignator>
31              </Match>
32              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                    >
33                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                    ">open</AttributeValue>
34                <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:
                    action:action-id" Category="urn:oasis:names:tc:xacml:3.0:
                    attribute-category:action" DataType="http://www.w3.org/2001/
                    XMLSchema#string" MustBePresent="true"></AttributeDesignator>
35              </Match>
36            </AllOf>
37          </AnyOf>
38        </Target>
39      </Rule>
40      <Rule Effect="Permit" RuleId="basic-customer-rule-latest">
41        <Target>
42          <AnyOf>
43            <AllOf>
44              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
45                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                    ">basic</AttributeValue>
46                <AttributeDesignator AttributeId="subject:customer-role" Category
                    ="urn:oasis:names:tc:xacml:3.0:attribute-category:subject"
                    DataType="http://www.w3.org/2001/XMLSchema#string"
                    MustBePresent="true"></AttributeDesignator>
47              </Match>
48              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                    >
49                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                    ">latest</AttributeValue>
50                <AttributeDesignator AttributeId="resource:resource-catalogue"
                    Category="urn:oasis:names:tc:xacml:3.0:attribute-category:
                    resource" DataType="http://www.w3.org/2001/XMLSchema#string"
                    MustBePresent="true"></AttributeDesignator>
51              </Match>
52              <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal"
                    >
53                <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string
                    ">open</AttributeValue>
54                <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:
                    action:action-id" Category="urn:oasis:names:tc:xacml:3.0:
                    attribute-category:action" DataType="http://www.w3.org/2001/
                    XMLSchema#string" MustBePresent="true"></AttributeDesignator>
55              </Match>
56            </AllOf>
57          </AnyOf>
58        </Target>
59        <Condition>
```

```
60          <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:integer-less-
                than">
61              <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#integer">
                    1</AttributeValue>
62                  <AttributeDesignator AttributeId="resource:
                        basicCustomersAccessingVideo" Category="urn:oasis:names:tc:
                        xacml:3.0:attribute-category:resource" DataType="http://www.
                        w3.org/2001/XMLSchema#integer" MustBePresent="true"></
                        AttributeDesignator>
63          </Apply>
64       </Condition>
65    </Rule>
66    <Rule Effect="Deny" RuleId="defaul-deny-rule"></Rule>
67 </Policy>
```

Listing 8.1: XACML policy example for the streaming video service scenario.

In this scenario, the adoption of the proposed blockchain based Access Control service has multiple advantages:

- relieving the content provider from deploying and managing its own access control system;
- guaranteeing both premium and basic customers that the access control policy actually enforced is really the one declared by the content provider (since customers can easily inspect the enforced policy by accessing the blockchain at any time);
- guaranteeing each basic customer that, in case their access to a latest content is denied, another basic customer was accessing the same content at that time, i.e., the access to such content was not unduly forbidden to them (in this case too, the basic customer can inspect the transactions recorded on the blockchain to verify the reason that lead to the access denial);
- guaranteeing all users to be charged by the content provider only for the contents they actually accessed, because all the access requests and the related access decisions are permanently recorded on the blockchain.

In the second reference application scenario, instead, the resources that are protected through the proposed access control system are smart contracts executed on the blockchain as well. We refer to this second scenario as *fully blockchain scenario*.For the sake of clarity, in the following we will call these smart contracts SMART RESOURCES. We do remark that no assumptions on the SMART RESOURCES is made, i.e., they can be contracts of any kind whose execution, for reasons that are immaterial here, needs to be protected through an access control system.

Of course, such contracts could contain a much sophisticated logic to monitor the accesses (e.g. elaborating statistics about the accesses and requests, load balancing, etc.), where the access control task is just a security component.The proposed access

control service implements the access control part of the SMART RESOURCE and it is independent from the specific operations it implements. This is the reason why, to both remain agnostic from the smart contract purpose and to stress out that it plays the role of the resource in this scenario, we named it SMART RESOURCE.

8.2 Blockchain-based Access Control Service

This chapter proposes a novel and alternative approach to implement Access Control Services exploiting blockchain technology. The basic idea underlying our approach is to exploit a blockchain to store access control policies and manage attributes, as well as to execute the access decision process, i.e., to evaluate the relevant policies exploiting the required attributes every time an access control request is issued by an user who wants to access a resource. A simple solution is the one proposed in [84] and shown in Chapter 7, where the XACML policy (represented in a more compact form, or even referenced) is stored directly on the Bitcoin blockchain. In this chapter, instead, we overcome the limitation of the previous chapter by leveraging smart contracts capabilities to define a more powerful solution. In particular, we represent an access control policy through a smart contract, called SMART POLICY, which is stored on the blockchain with a proper transaction, called POLICY CREATION TRANSACTION (PCT), when the resource owner creates and deploys it. Since the blockchain is an append only ledger, once uploaded, a SMART POLICY will be stored on the blockchain forever. However, the SMART POLICY can be replaced by simply uploading a new one, or even disabled with a proper transaction, called POLICY UPDATE TRANSACTION (PUT). The access decision process leverages the blockchain as well. In fact, each time a subject issues an access request, a proper transaction on the blockchain, called POLICY EVALUATION TRANSACTION (PET), is created by our service. This transaction includes a reference which causes the evaluation of the SMART POLICY and the production of the related access decision (e.g., Permit or Deny). The evaluation of such policy is completely executed on the blockchain, as we detail in the following.

In the rest of this section, we describe in details how we represent access control policies, create and store them in the blockchain, revoke or update them, and evaluate them by retrieving the required attributes to produce an access decision. Since the proposed system is based on the XACML standard, we describe how the architectural components in charge of the previous tasks according to the XACML reference architecture (see Figure 6.1) are defined on top of blockchain technology in the proposed system. The resulting architecture is shown in Figure 8.1.

The proposed system can be easily adopted in the reference application scenarios presented in Section 8.1, as detailed in Sections 8.4 and 8.5. Obviously, the kind of resource to be protected influences the structure of the PEP deemed to interact with it, and of other components of the system. Nevertheless, some components of

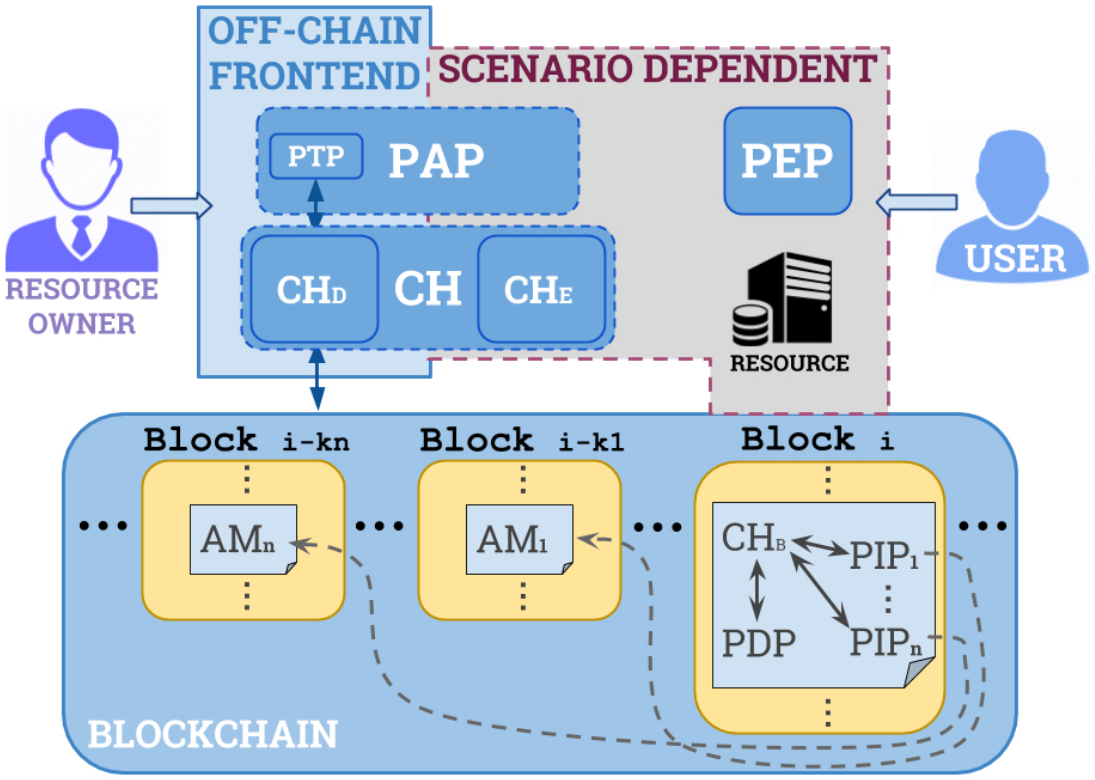


Figure 8.1: Architecture of the blockchain Based Access Control service.

the architecture are only consequence of our approach and so their implementation does not depend on the application scenario, as we will show in the following.

Moreover, we do note that the architecture of our blockchain based Access Control system is independent from the underlying blockchain technology chosen, provided that such blockchain supports smart contracts. However, the proposed implementation is based on a specific blockchain, Ethereum. Although the usage of our system does not require technical knowledge of the underlying blockchain technology, further data could be required to use it depending on the blockchain chosen. For example, in an Ethereum style blockchain, gas needs to be paid to use it (see Section 1.6) and so a user that wants to use such a system needs to own a wallet with some funds on it. This means that users, to use the proposed system, need to provide also some (not sensible) wallet information, and they will be required to perform additional operations (e.g., signing blockchain transactions).

8.2.1 Smart Policy

The solution we propose in this chapter is focused on ABAC policies, although we think that it could be easily extended to cover other access control models. An ABAC policy consists of a set of rules (see Section 6.1) that are combined

exploiting proper combining algorithms (e.g., simple logical operators such as AND, OR, or other algorithms) and they must be satisfied accordingly in order to grant the requested access. For the policy writing, we adopt XACML because it is a very expressive language allowing to write complex ABAC policies. Moreover, since it is a well known standard, some tools for policy editing and management are available both from academic and business organizations. In our approach, an XACML policy is properly translated into a smart contract, the SMART POLICY, in order to store it on the blockchain and execute it when necessary. Hence, the SMART POLICY can be seen as an executable version of the XACML policy. In other words, following the XACML naming, we could say that the SMART POLICY embeds a Policy Decision Point (PDP) customized for the execution of a specific access control policy. In the following we will see that the SMART POLICY also embeds the Policy Information Points (PIPs) required to interact with the SMART AMs.

8.2.2 Smart AMs

The attributes representing the features of subjects, resources and environment are stored on the blockchain as well, and they are implemented by a set of proper smart contracts. In this way, we exploit the blockchain advantages also for attribute management. In fact, the values of attributes and their updates are not alterable and auditable, since the values are stored on the blockchain, and their updates can be executed only through blockchain transactions, which are recorded on the blockchain as well. Following the XACML naming, the entities which created those contracts are the Attribute Providers, and the contracts which store the set of attributes can be seen as the Attribute Managers of such Attribute Providers. Hence, we will call such smart contracts SMART AMs. In our proposal we assume the existence of an ecosystem of SMART AMs deployed, maintained and advertised by third parties. For example, the smart contract of an institution could offer public information about its employees (e.g., their roles). These SMART AMs could also require some way of payment (either in or off chain) for the use of their services, so a market of AMs could naturally emerge. A SMART AM is invoked by a SMART POLICY to retrieve the current values of the attributes it manages. The part of the SMART POLICY which invokes the SMART AMs could be seen as the Policy Information Points (PIPs), i.e., the components of the XACML reference architecture devoted to the management of the attributes required for the evaluation of the policy.

8.2.3 Smart Policy creation

The process to create a new SMART POLICY is divided in two steps: the policy translation from XACML to smart contract, and the deployment of such contract on the blockchain. The first operation regards policy management and so it is tasked to the PAP. The SMART POLICY deployment requires an interaction between

the PAP and the blockchain and so is delegated to a part of the CH, the Deployment CH (CH_D).

The lifecycle of a SMART POLICY starts when the resource owner writes a new XACML policy to define the access rights on their resource(s) and submits it to the PAP. The Policy Translation Point (PTP) is a module of the PAP which translates the logic expressed by the XACML policy into a smart contract, the SMART POLICY. The SMART POLICY is not a simple rewriting of such policy, but it contains all the logic for its execution as well. For instance, each XACML statement referring to an attribute is translated inserting into the smart contract a function call to retrieve the current attribute value from the corresponding SMART AM each time the SMART POLICY is invoked. This is possible because in our approach the Attribute Managers are represented as smart contracts stored on the blockchain as well (as shown in Section 8.2.2). Furthermore, the SMART POLICY also encapsulates the logic for the policy evaluation process. Summarizing, codifying a policy as a SMART POLICY allows us to write blockchain executable policies from XACML ones. Such SMART POLICIES perform the tasks that in traditional Access Control systems are delegated to the PDP and PIPs. In fact, the decision process of the PDP is now performed through the smart contract execution, and the attribute values retrieval process of the PIP is achieved through smart contract function calls.

Once translated, the SMART POLICY is deployed on the blockchain by the CH_D, which issues the PCT. The expenses related to the SMART POLICY deployment are paid by the policy creator. This is correct since the policy creator is the entity that benefits from having an access control on the resource.

8.2.4 Smart Policy revocation and update

Policy owners can decide at any moment to revoke their SMART POLICIES. This is achieved by adding a self destruct operation function inside the smart contract representing the policy. The contract also enforces the constraint that only the resource owner (i.e., the creator of the contract) is allowed to call this function. Since the resource owner is the one issuing the revoke operation (creating the relative blockchain transaction), they are also the one paying the price of revoking a policy.

Do note that in most blockchain technologies available today, the blockchain is immutable and so the smart contract is not actually removed from the chain. Note that this would allow auditability, in fact, in such cases it is instead marked as not callable and so future calls to that contract will fail as expected, but the actual contract data remains publicly visible in the chain.

Updating a SMART POLICY means changing the related smart contract. Again, blockchains do not usually allow to change a smart contract code. So, updating a smart contract simply means deploying the new smart contract and use its new address instead of the previous one everywhere needed.

8.2.5 Smart Policy evaluation

The SMART POLICY evaluation is executed every time a subject tries to access the protected resource causing the PEP to issue the transaction that triggers the SMART POLICY contract execution (PET).

For the sake of simplicity, here and in the following we say that "the SMART POLICY evaluation is executed on the blockchain" meaning that this SMART POLICY execution is replicated among the miners elaborating the new block to be added to the blockchain.

To trigger the SMART POLICY execution, the PEP creates the access request and sends it to the Evaluation CH (CH_E), which, in turn, creates the proper transaction, PET, to trigger the evaluation of the policy and submits it to the blockchain.

The evaluation transaction triggers the execution of the main method of the SMART POLICY, which implements some of the task of the CH, i.e., it coordinates the execution of the phases of policy evaluation. In particular, the updated value of all the attributes required for the evaluation of the policy are collected when needed. This phases are executed by the functions of the SMART POLICY which implement the PIPs. These functions issue a number of messages triggering the execution of the SMART AMs corresponding to the required attributes. The function corresponding to the PDP tailored for the specific policy is executed with the retrieved attribute values. This produces the decision that will be returned as result of the policy evaluation. Do note that attribute values are retrieved in a lazy way, i.e. they are only retrieved when needed for a function evaluation. This is different from a traditional system where all attribute values are usually obtained during an initial separate step, before evaluating a policy. We choose the lazy solution to try and minimize the number of external calls executed by the SMART POLICY. In fact, in general, each attribute is retrieved via a call to a different contract (the corresponding SMART AM) and this is an expensive operation (in terms of fees consumed) that is completely useless if the value is not actually used. Lazy attribute values retrieval allows for a cheaper and faster policy evaluation.

The expenses of the evaluation transaction are paid by the subject making the access request (since the subject will be the entity benefiting from the granted access). This prevents subjects from spamming requests to the system, since they are limited by the value they own. Hence, the subject needs to manage a wallet holding value on the underlying blockchain and it is required to interact with the access control system (e.g. for signing transactions) to validate the payment of the required expenses.

8.3 Reference Implementation Tools

In order to validate and evaluate the proposed approach, we have developed a proof of concept implementation of the blockchain based Access Control System

presented in this chapter for both the reference examples depicted in Section 8.1. We will present both implementations in the relevant sections (Section 8.4.3 and Section 8.5.1), while in this section we depict only the common tools and components used by both.

8.3.1 Chosen Tools

To implement our system we have chosen the Ethereum blockchain protocol (as of December 2017), because it is strongly focused on smart contracts and because it is nowadays a widely used smart contract ready blockchain protocol proposal. We then chose Solidity (see Section 1.6) as programming language to write the smart contracts and java to write the off-chain side of our framework.

To deploy and test our system, we used the *International Educational blockchain* academic testnet (part of the *Open Blockchain* initiative [203]). This is a Ethereum based private testnet with nodes currently run by North American and European universities, and it allowed us to have an environment at the same time controlled and somewhat realistic. Due to the small size of the community currently using this testnet it provided us with a perfect simulation tool to deploy our system (e.g. we could artificially influence parameters, like block congestion, as required), but it lacked the randomness and practical issues of a real widely used network, running several different contracts all the time. This is way we also used the Ethereum official testnet *Ropsten* [233] for experiments, to obtain results on a global and more realistic testnet. In fact, Ropsten is the official PoW Ethereum testnet, revived in March 2017 after a DoS attack [232]. Since it uses PoW like nowadays Ethereum (despite the proposals to move past it) it better models the real network compared to the other two official testnets, *Rinkeby* [227] and *Kovan* [159], that both use PoA (i.e. Proof of Authority) to be protected against attacks like the aforementioned one that affected Ropsten at the beginning of 2017. To access both testnet blockchains we used `geth` [125], one of the most used Ethereum clients. We did not use the Ethereum main chain because of the obvious cost constraints, and also to avoid to burden the immutable Ethereum main chain with our test data that are intended to be temporary.

To allow our java client to interact with `geth` we use the `web3j` [251] Java library. `web3j` is a lightweight library that, among its many functionalities, supports all of the JSON-RPC API offered by `geth`. It also allows automatic creation of Java smart contract function wrappers from Solidity ABI files (see Section 1.6). These functionalities provide us with the needed tools to bridge the blockchain components of the system with the off-chain components.

8.3.2 Smart Policy Translation

We have shown in Section 8.2 how the SMART POLICY concept is shared by both our reference scenarios. We will now show how this common concept is implemented.

The component tasked with SMART POLICY management is the PTP (see Section 8.2.3). The PTP is written in java and its task is to translate an XACML policy created by the resource owner in a Solidity smart contract encoding the SMART POLICY. As a matter of fact, the PTP embeds a XACML parser that translates the policy into a smart contract written in Solidity.

The smart contract generated contains some utility functions which are the same for all policies (e.g., a suicide function that is invoked to revoke the SMART POLICYas explained in Section 8.2.4) and data (e.g., the address of the resource owner, marked as private field). However the main function of the SMART POLICY is called `evaluate` and represents the executable version of the XACML policy. In the following, we show how the XACML policy is translated in Solidity to produce the body of the `evaluate` function.

An XACML policy consists of a policy Target and a set of rules, each including their Targets and Conditions [200]. We focus our description on the translation of the Targets and rules of the policy, being the translation of the Conditions very similar. A Target is a combination of <`Match`>..<`/Match`> elements, each element will be called MATCHE in the rest of this chapter. Each MATCHE is translated as a check of the `evaluate` function. The type of the Solidity function to be used to implement the check is derived from the XACML `MatchId` field and the data type from the XACML `DataType` field of <`AttributeValue`> and <`AttributeDesignator`>. Each check, to be performed, needs the current value of an attribute at access request time. Hence, the SMART POLICY must be also able to retrieve these values in order to compute the decision result. This would be a task of the PIPs in a traditional XACML system, while, in our proposal, we integrate the PIPs functionalities in the SMART POLICY through calls to SMART AMs(see Section 8.2.2). Any resource owner who creates a new policy needs to specify in such policy the AMs to be interrogated to retrieve the required attribute values. To this aim, for each MATCHE the resource owner chooses the SMART AM to be called by specifying in the <`AttributeDesignator /`> tag the SMART AM function name through the `AttributeId` field and the SMART AM address through the `Issuer` field. All the MATCHE checks boolean results are properly composed with conjunctions or disjunctions as defined by the policy through the tags <`AllOf`> and <`AnyOf`> respectively.

Finally, the `evaluate` function needs to return the result of the decision process. One solution would be to simply save the result as data in the state of the contract, but, instead, we opted for firing an event containing the request id (i.e., the id of the transaction encoding the access request) paired with the corresponding result. Events are data saved on the EVM log instead of the contract storage space, exploiting them to return the result of the evaluation function is a cheaper way of storing the decision for every request, at the expense of making such decisions invisible ad so unusable to the contracts. In our current system this limitation is not a problem, but this approach could of course be changed if needed.

We make now an important remark about the parser. Since XACML derives

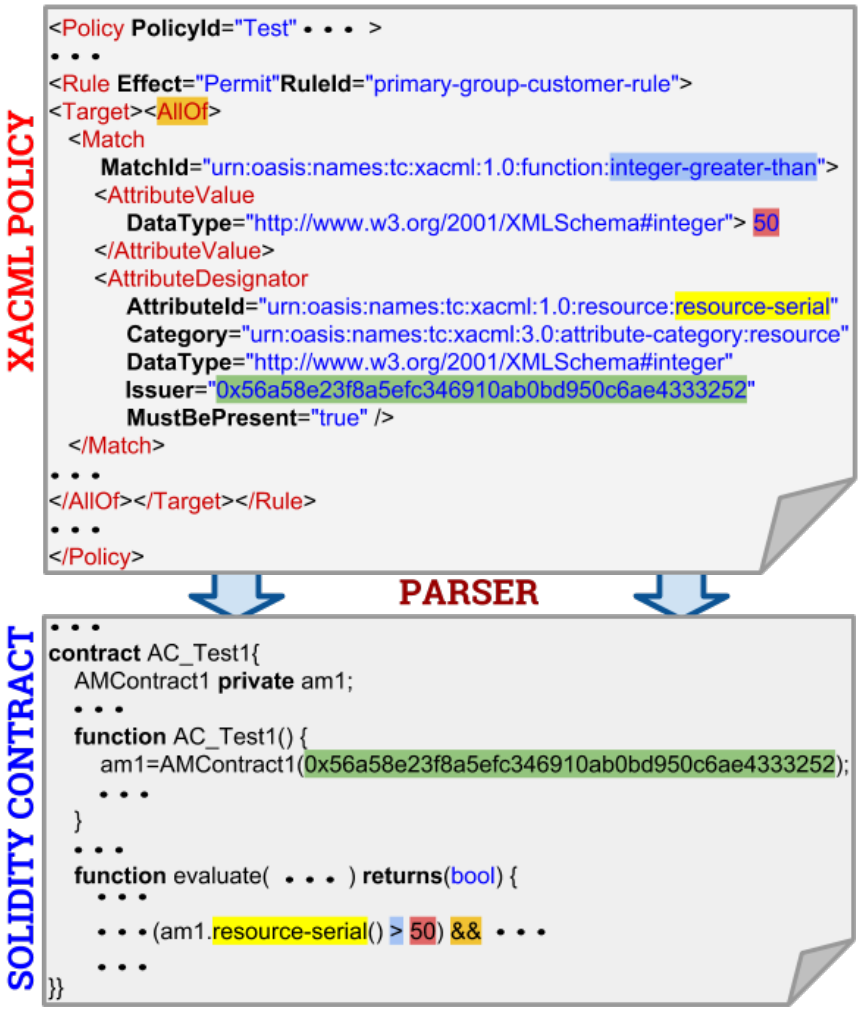


Figure 8.2: Simplified XACML to Solidity parser example.

from XML, the parser is implemented as a classical XML parser. As such it has linear computational complexity. Nevertheless it is too resource intensive to be executed on the Ethereum blockchain. The parser execution would certainly cause an out of gas exception (i.e. when the execution exceeds the block gas limit) for any but the most trivial policy. This is assuming that the smart contract to execute the parser could itself be deployed without incurring in an out of gas exception. Even if we could deploy and execute the parser as a smart contract, passing it an argument (i.e. an XACML policy) for execution would be prohibitively expensive. XACML is a very verbose language while message size is a scarce and priced resource on the blockchain. All and all, the fact of executing the parser on the blockchain would not yield any real advantage while remaining indubitably cumbersome for the whole blockchain community and expensive for the single user. We will show later that the user can check the parser result (i.e. the SMART POLICY code) before deployment,

so any error or fraudulent behaviour can be detected and thwarted.

8.4 Controlling the Access to a Video Streaming Service

The first reference application scenario of this chapter concerns the regulation of the access to a video streaming service (see Section 8.1). Hence, the resource to be protected is the video streaming service, which must be properly instrumented to embed the PEP functionality. This could be simply executed by deploying the PEP as a proxy service of the real one. In this way, the PEP would intercept all the requests to the service because the service cannot be reached directly, i.e., the PEP must be the only one allowed to interact with the service. Hence, the users of the video streaming service will interact with the PEP, which exposes the same interface as the video streaming service, and which will interact with the Frontend of the blockchain Based Access Control Service. The service frontend includes the enforcement CH, i.e. the CH_E, which is in charge of crating and submitting the PET to the blockchain in order to trigger the execution of the decision process. Only in case of positive decision, the PEP will execute the requested operation by forwarding the access request of the user to the real service. The architecture of the proposed system customized for the first reference application scenario is depicted in Figure 8.3 (do note how the scenario dependent components of Figure 8.1 have been consequently instantiated).

8.4.1 New policy creation

The first step of the system life cycle is the policy creation. As explained in section Section 8.2.3 the video streaming service owner submits the XACML policy he wants to be enforced on his service to the PAP. In the current scenario, the PAP is a service as well, which runs within the blockchain Access Control Service Frontend. The PAP embeds the PTP, which translates the XACML policy to a SMART POLICY, and such smart contract is passed to the Deployment CH, i.e. the CH_D, to be compiled and deployed on the blockchain.

Finally, when the SMART POLICY gets accepted by the blockchain, a new entry is created by the PAP on the SMART POLICIES Table (SPT). This entry will store the address (or any other form of contract linking provided by the specific blockchain technology adopted) of the SMART POLICY paired with the ID of the video streaming service. In this way, the right policy can be retrieved at access request evaluation time for each resource (i.e. video streaming service).

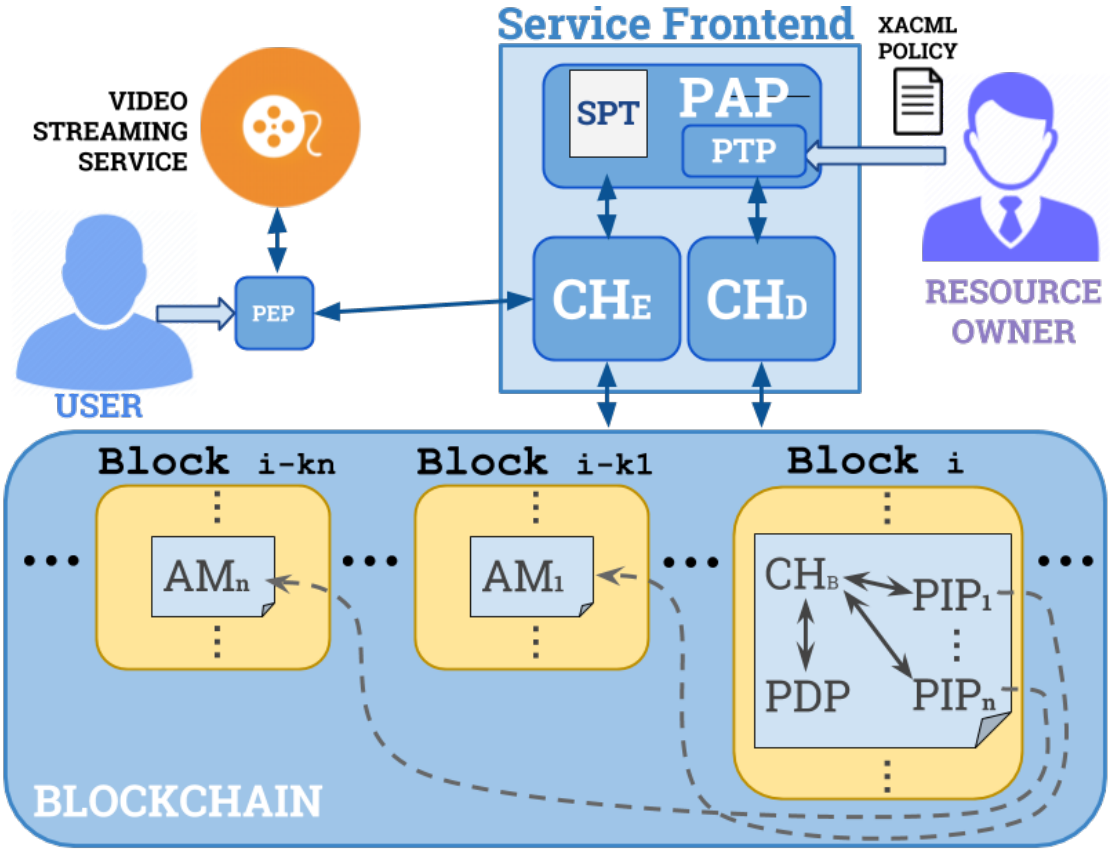


Figure 8.3: Architecture of the blockchain Based Access Control service for the first reference scenario.

8.4.2 Access request evaluation

Once a SMART POLICY has been deployed on the blockchain, we can say that the blockchain based Access Control service starts waiting for access requests. Hence, when the PEP intercepts an access request, it forwards it to the CH_E, which is the part of the CH in charge of bridging the access request with the underlying blockchain. The CH_E queries the PAP sending it the ID of the resource (i.e. video streaming service) in the access request to identify the SMART POLICY to be invoked on the blockchain. The PAP returns the address of this SMART POLICY by accessing the SPT exploiting the ID of the resource as index to locate the right entry. In order to trigger the execution of such SMART POLICY, the CH_E creates the related PET from the access request. This transaction will be processed by the blockchain triggering the SMART POLICY to be executed (possibly executing other smart contracts of the AMs needed) and producing a Permit or Deny result. The CH_E reads back from the blockchain the decision and informs the PEP that then will actually enforce the decision by granting or denying the request accordingly.

8.4.3 Proof of Concept Implementation

We note that our system implements many utility functionalities for the system manager and the users (for example to obtain the list of the last n submitted access requests), but in the following we will only focus on the two main operations of new policy creation and access request, due to space constraints.

PEP

The PEP is written in java and it is the component of the system interacting with the protected resource. Our system is designed as a program pluggable to already existing Access Control scenarios. This is why the PEP is inserted in the access interface of the resource and provides an API with a method to submit an access request (written in XACML) which returns a response (written in XACML as well). The PEP was left intentionally lightweight since its main task is to bridge our system with the resource and subjects. The logic functionalities of blockchain client are delegated to other internal components. This is why the PEP only forwards the received requests to the CH_E, possibly enriched with additional information, if available.

PAP

The PAP is written in java and it is the software component in charge of policy management and retrieval. Its main tasks are to transform an XACML access control policy written by the resource owner into a SMART POLICY, and to remember the mapping between SMART POLICIES and resources.

The PAP receives new policies from resource owners and leverages the PTP module explained in details in Section 8.3.2 to translate them into SMART POLICIES. Once the PTP has finished creating the contract, the PAP sends it to the CH_D to be compiled and deployed on the blockchain, and waits for an answer from the CH_D. If the contract deployment phase from the CH_D (see later) is successful, the CH_D communicates to the PAP the address of the newly deployed contract. The PAP then stores in the SPT a new entry consisting of the resource ID (i.e. video streaming service ID) and the received address.

CH

The off chain sides of the CH (i.e., CH_E and CH_D) are components written in Java, and they have the task of managing the access to the blockchain on behalf of the PAP and the PEP. At policy creation time, when the CH_D receives from the PAP a SMART POLICY written in Solidity it compiles the Solidity code to EVM bytecode (see Section 1.6) using the `solc` compiler [246]. The CH_D then uses `web3j` to wrap it into a transaction for the deployment on Ethereum through the `geth` node. At this stage the CH_D can optionally perform additional checks on the contract deployment

transaction. For example, it could query the blockchain (using the `geth` node) to check whether the policy creator has enough credit (i.e., ether) in its account to pay for the expected gas cost, or it could check whether the SMART AMs invoked by the SMART POLICY do actually exist on the chain.

An important step is that the contract deployment transaction needs to be signed by the user who is paying for it. In particular, once the transaction is ready to be signed, it is made visible to the resource owner who can check it (possibly with an automatic tool), sign it, and then communicate the signed transaction back. In our implementation the user interacts with the system through an interactive interface where it is first required to insert the policy it wants to add, then it is informed of the derived smart contract and relative deployment transaction waiting to be signed. The user can then check that the transaction correctly represents the policy it intended and, if it is satisfied, communicate the signed version of the transaction. Once the CH_D receives the signed transaction it checks the signature, and if it is correct it sends it to the `geth` node to broadcast it to the network. The user receives a confirmation or error message depending if the deployment was successful or not. This approach guarantees that the private information about the users wallets are not disclosed to our framework. Once the transaction is actually inserted by a miner in a block, i.e., the SMART POLICY is on the blockchain, the CH_D receives back from `geth` the contract address, which is returned to the PAP to be stored in the SPT.

At access request time, the CH_E receives an XACML access request from the PEP. The CH_E parses such request to retrieve the resource ID (i.e. video streaming service ID) and the other attribute values, and it queries the PAP to retrieve the address of the SMART POLICY paired with that resource.

Then, the CH_E wraps the request into a transaction encoding a call to the `evaluate` function of the SMART POLICY referenced by the address retrieved from the PAP. This transaction needs to be signed by the subject that submitted the access request (as explained in Section 8.4.2). This step exploits the same signing process detailed above but with the difference that the accessing subject, and not the resource owner, is the one in charge of checking and signing the transaction. Once the transaction is signed, it is passed by the CH_E to `geth`, that will broadcast it on the Ethereum network. Once the transaction is mined the `evaluate` function fires an event with the evaluation result. This event is read by `geth` that passes it to the CH_E which, in turn, communicates the result to the PEP.

PDP & PIPs

The traditional tasks of PDP and PIPs are merged together into the SMART POLICY. This contract is dynamically generated by the PAP from an XACML policy and, once deployed, resides on the Ethereum blockchain in EVM bytecode. As already stated before, the decision process of the PDP is performed by the decentralised execution of the `evaluate` function of the contract and the attribute value retrieval is performed by internal calls of the contract directly to SMART AMs on the same

chain. All the communication is achieved through smart contract function calls and event firing that are implicitly managed by the Ethereum protocol, so we can say that the SMART POLICY also performs some CH tasks (i.e., CH_B in Figure 8.3).

8.5 Controlling the Access to Smart Contracts

This section describes the customization of the proposed blockchain based Access Control Service to be adopted in the second reference application scenario of Section 8.1. In this case, the resource to be protected is a smart contract, called SMART RESOURCE, and the resource owner is the author of such contract, who wants some *critical functions* of the SMART RESOURCE to be performed only by subjects holding the corresponding rights. In this scenario, the code implementing the PEP tasks is embedded in the code of the SMART RESOURCE itself. In particular, each of the functions of the SMART RESOURCE that need to be protected must invoke, as first operation, the evaluation of the access control policy, i.e., it must issue a message PET to trigger the execution of the SMART POLICY. Consequently, the address of the SMART POLICY must be directly embedded in the PEP that invokes a PET, thus playing part of the PAP role and making the SPT no longer necessary in this scenario. Hence, in this scenario, we can say that part of the PAP runs on the blockchain. The decision returned by the SMART POLICY will determine whether the actual code of the function will be executed or not.

The PAP tasks concerning the SMART POLICY creation, instead, are still performed by a service external to the blockchain, which runs within the blockchain Access Control Service Frontend, called PAP_O (i.e. off-chain PAP). This component is also in charge of inlining the PEP code, i.e., the creation and the invocation of the PET, into the SMART RESOURCE through a proper additional module called Resource Inlining Point (RIP). To this aim, the PAP_O requires in input both the SMART RESOURCE contract code and the corresponding XACML policy to be enforced. Then, the PTP produces the SMART POLICY as shown in 8.2.3, and it embeds the invocations to such policy in the SMART RESOURCE function through the RIP, so that they are both ready for being compiled and deployed on the blockchain by the CH_D. Of course the SMART POLICY needs to be deployed first to retrieve its address to be embedded in the modified SMART RESOURCE code. We also remark that the deployment phase can optionally be carried out directly by the resource owner instead of the CH_D, using our frontend only as a translation tool, if such approach is preferable. As we have already explained in Section 8.2, the tasks of the PDP and PIP are performed by the SMART POLICY in this scenario as well.

The resulting architecture is depicted in Figure 8.4. In principle, the PAP_O could also be implemented as a smart contract and deployed on the blockchain. This would leave no need for the CH_D and so the entire system would reside on chain. However, we already stated the unfeasibility of such approach in Section 8.3.2, mainly due to cost constraints.

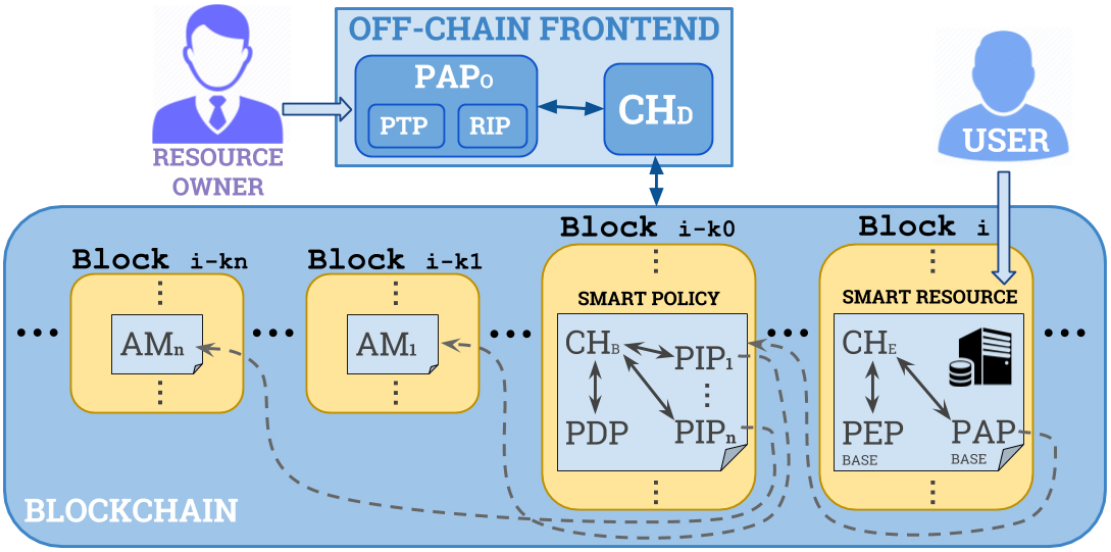


Figure 8.4: Architecture of the blockchain based Access Control Service customized for the SMART RESOURCE reference scenario.

8.5.1 Proof of Concept Implementation

We have developed a proof of concept implementation of the proposed system based on our reference implementation scenario (see Section 8.3.1) , i.e. developed on the Ethereum blockchain with smart contracts written in Solidity and off-chain side written in Java bridged to the blockchain by a geth node accessed through web3j.

Our aim is to provide a fully automated integration process of the XACML policy logic into the SMART RESOURCE without user intervention. To achieve this we assume that the provided XACML policy is written in a precise format.The policy needs to be a default (or non applicable) allow disjunction of rules, where each rule has a binding match node over the ACTION-ID attribute. Do note that we allow the owner to define a policy with more than one rule applicable for the same `ACTION-ID` attribute value. When an actual `ACTION-ID` value is passed at request evaluation time all the disjunct rules applicable are evaluated and the policy permits access accordingly. If none of the rules is applicable (because no rule was defined over that `ACTION-ID` value) then the policy resorts to a default permit. This solution allows the user to write a unique policy that contains rules for each different action on the resource that needs to be Access Controlled, leaving all the remaining unspecified actions not protected. This way we can allow the user to specify not only the conditions on a given action but also which actions need to be protected and which not. Do note that this approach is fully compliant with XACML philosophy and it is already used in traditional systems.

In our implementation we leverage the concept of *function modifiers* provided by solidity. In solidity a function modifier is used to wrap a function around a predetermined code. By creating modifiers containing checks at their beginning and halting execution in case of failure, through the *require(boolean function);* solidity construct, we can control a function execution. To this aim our service provides a fixed base contract called LINKER. This contract acts like a bridge between the SMART RESOURCE and the SMART POLICY. It contains a constructor requiring the address of the SMART POLICY as parameter to perform the linking between the policy and the resource. It then defines a modifier that is used to enforce the access request evaluation. This modifier simply calls the `evaluate` function of the SMART POLICY (similar to the previous implementation in Section 8.4.3) and halts the execution if it returns false. The SMART RESOURCE contract needs only to specify that it inherits from LINKER and add the modifier to the signature of every function, not just the *critical* ones (thanks to the required policy format expressed before).

$\mathbf{PAP}_O$ & $\mathbf{CH}_O$

In practice all the creation of the support smart contracts (such as LINKER) and their linking to the SMART RESOURCE during policy creation time are automated and transparent to the user. The user only has to pass to the PAP_O its input data (i.e. the SMART RESOURCE code in Solidity and the policy in XACML). The only requirement is that the `ACTION-ID` actual values specified in the policy should match with the corresponding function names of the resource contract that needs to be protected. The off-chain components of our system are only active at policy creation time (or revoke and update) and they execute the following operations:

- the XACML policy is parsed by the PTP to obtain the SMART POLICY code as explained in Section 8.3.2;
- the SMART POLICY is passed to the CH_D that deploys it on the blockchain after owner approval and signature (as already explained in Sec 8.4.3) and the corresponding contract address is obtained. This is the only needed information not known statically;
- the CH_D enriches the SMART RESOURCE code by adding the LINKER contract code (with the correct SMART POLICY address obtained after the previous step) and the inheritance clause in the current contract header. Then the LINKER function modifier is added to the signature of each public method;
- finally the enriched SMART RESOURCE code can be optionally automatically deployed on the blockchain by the CH_D with the exact same procedure used for the SMART POLICY deployment. Otherwise it can be communicated back to be deployed by the resource owner itself.

We remark that even if the function modifier to check for access rights is inserted for all actions it will only be pertinent for some ones, according to how expressed in the policy.

During request time the requested Access Rights evaluation is performed automatically by the blockchain protocol. Performing an action on the resource means calling a function of the deployed SMART RESOURCE. When this happens the function modifier inherited by LINKER is executed first, this triggers the execution and enforcement of the result of the `evaluate` function of the SMART POLICY. Only if the modifier execution is successful (i.e. only if the policy grants access for that action with the current access context) the action can be performed.

8.6 Considerations

The main advantage of our proposal is that the policy evaluation process is performed by smart contracts on a blockchain. This allows our decision system to inherit the blockchain technology advantages, i.e., always available, distributed (so no single point of failure or attack), tamper resistance, etc. An interesting property is auditability, which is derived from the immutability and transparency properties of blockchain technology. Since the smart contract execution is performed by the blockchain (i.e., replicated among the miners), it is beyond the control of both the resource owner and the subject making the request. So neither of them can forge a false decision. Moreover both the policy and the Permit or Deny evaluation result linked to the request identifier are stored on the blockchain as well, so they are both publicly visible. Any user whose access is fraudulently denied by the resource owner can prove that the access right should have been granted instead through the public data on the blockchain. Since the blockchain is immutable, this also holds for old SMART POLICIES. Even if a SMART POLICY has been revoked, its code and entire access request log remain still accessible in the blockchain.

Do note that in our hybrid scenario reference implementation (shown in Section 8.4.3) the resource owner could still fraudulently ignore access requests before any trace of them is left on the blockchain, by simply modifying the PEP to discard some of them. This issue could be avoided by having the resource owner advertise publicly the address of the SMART POLICIES allowing the subjects to directly send transactions to such contracts to obtain the access to the resources. This issue is of course not present in the fully blockchain scenario (shown in Section 8.5.1).

A clear consequence of auditability is a potential privacy issue. Some solutions to properly mask the publicly available information stored on the blockchain might be needed in a real world application.

One disadvantage of our proposal can be that blockchain technology is still a novel and emergent technology, and so most of the users have never used such technology, nor they have familiarity with the basic concepts of it. Although the proposed system automatically interacts with the underlying blockchain, the users

still need a basic understanding of the protocol, since they are required to own a wallet to sign and pay for transactions. Even if transaction checking and signing (see Section 8.4.3) as well as auditability checks described in this section are operations that can be performed automatically by a secure third party program, this would still require the user to trust such software. Hence the problem of trust would just be moved and not solved. The lack of user familiarity may be an issue for initial widespread adoption.

Another main concern about blockchain technology in general is performance. The need for a distributed consensus introduces an overhead non-existent in a centralised model where the system state updates are managed by a single (trusted) entity. Moreover, the replication of the shared state (i.e., the blockchain data) and replication of new data validation across all the nodes determines an additional burden for the participants. Because of these observations if we want to base our proposed system on a public blockchain we expect to incur in an inevitable loss of performances compared to a traditional centralised system, i.e., an increase in resource needed to run the Access Control system and an increase in the evaluation time to obtain an access decision. We also remark that, usually, public blockchain protocols introduce a cost per transaction (i.e., fees and gas). Of course this is only applicable to our hybrid scenario reference implementation, since in our fully blockchain proposal the user is already using smart contracts to define SMART RESOURCES. For the application to SMART RESOURCES already running on a blockchain the execution of other smart contracts as the same order of magnitude of complexity and so our system does not introduce such a notable overhead (both in time delays and resource needs).

To investigate this aspect, we evaluate the performances of our reference implementations in the next section.

Finally we want to note that the main contribution of our fully blockchain proposal basically consists in outsourcing the Access Control logic from the SMART RESOURCE. This means that the user relies on a third party service (i.e. our system) to write all the Access Control logic. But this does not introduce an heavy trust requirement in such third party. In fact, the service builds the SMART POLICY and integrates the SMART RESOURCE contract instead of the user, but it needs the user approval (i.e. its signature) to deploy both of them. This means that the user can check before deployment that the service was not malevolent and stop the deployment at any moment if it is not satisfied.

Such outsourcing of code writing grants a clear advantage in terms of user effort and errors avoidance. Since the SMART POLICY is automatically generated and linked to the SMART RESOURCE any human error is avoided. Furthermore, since the user does not have to write an Access Control contract custom implementation but relies on a statically written solution instead, checks of the contract security are easier and generalizable. Of course this also results in possibly fewer errors in the SMART RESOURCE implementation in general since the user can focus all its efforts and attention on its intended logic without minding about the Access Control side.

8.7 Experimental Results

To study the performances of our reference implementations we analyzed three measures:

- monetary cost;
- resource cost;
- time cost.

We do note that even the off-chain Frontend of our reference implementation is different from a traditional XACML Access Control System, mainly for the need for a XACML to Solidity parser. Nevertheless, the parser complexity is guaranteed linear in the size of the policy (and easily parallelisable in case of policy sets). We do not consider the parser during our performance evaluation, focusing only on the on-chain operations. In fact, on modern hardware, the parsing time is in the order of milliseconds even for big policies, while the on-chain operations take on average seconds to complete. Thus the different order of magnitude of the studied measures further justifies our simplification.

We point out that for both our reference example scenarios presented in Section 8.4 and Section 8.5 the SMART POLICY deployment phase is exactly the same. Consequently, the gas and time costs are the same as well. The SMART POLICY evaluation phase is, instead, different. In fact, in the traditional digital resource scenario, a transaction calling the `evaluate` function of the SMART POLICY is submitted to the blockchain, while in the SMART RESOURCE scenario the `evaluate` function is called by the SMART RESOURCE itself, through a smart contract message. However, the gas cost required for the execution of the `evaluate` function is the same in both scenarios, the only difference is that in the traditional digital resources scenario the transaction cost needs to be paid explicitly (for further details see Section 8.7.1), while in the SMART RESOURCE case this cost was covered already by the SMART RESOURCE caller. This means that, from the gas cost point of view, the gas price payed by an instance of the first scenario is an upper bound (i.e., the SMART POLICY cost plus the transaction cost) for the second scenario (SMART POLICY cost only), obviously, considering the same SMART POLICY. The `evaluate` function time cost is the same in both scenarios as well. However, in the SMART RESOURCE scenario, the SMART RESOURCE and SMART POLICY evaluations are inserted in the same block, and this would mitigate the time cost of the SMART POLICY. For the previous reasons, in the following, we evaluate SMART POLICIES deployment and execution costs (considering either gas or time) independently from the scenario chosen.

```
1 pragma solidity ^0.4.18;
2 contract AMContract {
3     function get_value() public constant returns (uint32);
4     function get_boolValue() public constant returns (bool);
5 }
```

```
6  contract PDP_PolicyDecisionPoint {
7  AMContract private am0;
8  event Evaluation(
9      address indexed _from,
10     uint32 _id,
11     bool _decision
12 );
13 address private owner;
14 modifier onlyOwner {
15     require(msg.sender == owner);
16     _;
17 }
18 function close() public onlyOwner {
19     selfdestruct(owner);
20 }
21 function PDP_PolicyDecisionPoint() {
22     owner = msg.sender;
23     am0 = AMContract(0xe5b21187a7d5ca08f02dfcf047ab9f9d52c1f455);
24 }
25 //unique nonce is passed
26 function evaluate(bytes32 subject,uint32 nonce) returns(bool) {
27     bool decision = (am0.get_value()>0);
28     Evaluation(msg.sender, nonce, decision);
29     return decision;
30 }}
```

Listing 8.2: Simplified reference SMART POLICY with a single SMART AM and a single MATCHE.

8.7.1 Monetary Cost

Using a fee (or gas) based blockchain, we introduce a monetary cost for every transaction that is mined. In particular, since our reference implementation is based on the Ethereum protocol, in which gas is consumed by transactions, we performed a set of experiments to estimate the gas cost of our two kinds of transactions: SMART POLICY deployment and evaluation.

smart policy deployment transactions

The gas cost of a transaction deploying a new contract (Gas_D) can be expressed as follows:

$$Gas_D = FixedCost_D + CodeCost + InitCost$$

where $FixedCost_D$ represents the fixed amount of gas that must be payed independently of the code of the contract to be deployed (e.g. the fixed cost to create a new transaction, 21 000 gas at the time of writing, and the fixed cost to deploy a new contract, 32 000 gas at the time of writing [276]), $CodeCost$ represents the cost to store the actual contract (i.e., the code) on the blockchain, and $InitCost$

represents the computational cost incurred to run the constructor instructions and initialize the contract. In our implementation each contract representing a policy has a fixed core of utility methods and variables that contribute as a constant amount to both *CodeCost* and *InitCost*. For instance, to save the address of the contract creator in the constructor requires a store operation (which currently costs 20 000 gas) that contributes to *InitCost*, while adding functions to revoke the SMART POLICY increases *CodeCost* because it increases the code length. The policy dependent contributions to *CodeCost* and *InitCost* are mainly due to the number of rules of the policy, to their complexity, and to the number of different SMART AMs they require to contact.

Obviously, more rules and more complex rules require more code to be stored and managed. Since each rule consists of a set of MATCHES elements, we measure the complexity of a policy as a function of the number of MATCHES and of their complexities. Less obvious is the contribution of the number of SMART AMs. This is due to the fact that the SMART POLICY stores the addresses of the SMART AMs needed to retrieve required attributes. These addresses are known at deployment time and are saved in contract variables by the constructor, so each address to be remembered causes a costly store operation. Do note that the internal complexity of the SMART AMs does not influence the deployment cost of a contract requiring them (but it will influence, instead, its execution cost). In our test we experienced that to deploy an empty policy (i.e., a policy which always returns Permit), we consumed about 175 000 gas, while to deploy a policy with one simple rule performing the comparison of the value of one attribute with a constant (i.e., invoking one SMART AM) we consumed about 280 000 gas. Allowing the policy to use an additional SMART AM (to get the attribute values to be used in the MATCHES) consumes approximately 26 000 gas alone (to store the AM information), and each additional simple MATCHE in the policy consumes about 46 000 gas (but complex MATCHES may consume more gas). These are very rough estimations, and should only be considered as a lower bound of the actual cost. Knowing that the current gas limit in our academic testnet is about 4 700 000 for each block, it is possible to estimate the maximum size (i.e. number of different SMART AMs and MATCHES) of a deployable policy in our testbed. According to our rough estimation, for example, a policy using 10 different SMART AMs and 90 MATCHES would have about the maximum size that could fit in one block. We think that a policy of that kind is quite large, because common policies typically use two or three attributes, and we do remark that this is just a constraint of our reference implementation and not of the proposal itself.

smart policy evaluation transaction

To trigger the evaluation of a SMART POLICY for a given access request we use a transaction calling the `evaluate` function of the contract (See Section 8.2.5). The

gas cost of such transaction (Gas_E) can be expressed as follows:

$$Gas_E = FixedCost_E + EvalFunctionCost$$

Where $FixedCost_E$ represents the fixed cost of the transaction performing the call (and carrying the function parameters), and $EvalFunctionCost$ represents the cost to execute the `evaluate` function. As explained in Section 8.3.2, the `evaluate` function is a combination of boolean functions representing a MatchE each. Furthermore, each encoded MatchE usually invokes one (or more) smart AM to retrieve attribute values. This means that the cumulative evaluation cost depends not only on the number of MatchEs and their individual complexity, but also on the complexity of the smart AMs invoked. The gas cost estimation is further complicated by the use of short circuiting logical operations. In fact, the execution of the same expression could have very different costs depending on the actual values of attributes at execution time. To test this, we performed some experiments where we evaluated a policy using 3 different smart AMs and 80 MatchEs in conjunction which exploit the values of the attributes provided by such smart AMs. The first time we purposely choose attribute values to satisfy all the MatchEs, obtaining a Permit result and consuming 210 643 gas for the relative transaction. Instead, setting the attribute values in such a way that the first MatchE is not satisiefd, the entire expression short circuits to false without evaluating all the remaining MatchEs. This results in a Deny decision consuming 32 267 gas only for the relative transaction. The second execution consumed approximately 15% of the amount of gas consumed by the first execution. Moreover two thirds of this cost was due to the fixed cost of the transaction itself (more than 20 000 gas). Hence, considering only the cost due to the $EvalFunctionCost$, the second execution costs about 5.6% compared to the first.

Due to all this, it is difficult to get a general estimation of the cost of the evaluation function. To give an estimation of the gas cost of an evaluation, we deployed a smart policy consisting of 90 MatchEs referencing 10 smart AMs. The policy was a conjunction of simple boolean conditions that we knew being all true with the values returned by the smart AMs (in order to avoid short circuiting). The resulting cost of an evaluation transaction (returning a Permit) under the previous assumptions is of approximately 230 000 gas. This shows how the policy evaluation cost (in the worst case that all MatchEs need to be executed) is considerably lower than the initial policy deployment cost. For example, for the policy of the above example, the evaluation transaction cost is about 5% of the gas consumed by the corresponding deployment transaction. Of course, this is just a rough estimation which depends on the policy an on the smart AMs chosen. For instance, using a very costly smart AM arbitrarily increase the policy evaluation cost without influencing the initial deployment cost (that is independent of it).

Given a reference policy (see Listing 8.2) invoking a single smart AM and consisting of a conjunction of n simple single attribute MatchEs crafted to be

always true over the SMART AM returned values, we depicted in Figure 8.5 the deployment and execution costs for increasing values of n.

We observe in Figure 8.5 that both the deployment and execution cost are linear in the number of MATCHES (of the same complexity), and that the deployemnt cost is much higher than the execution cost. This is a desirable property since the deployment cost needs to be paid only once, while the executions cost needs to be paid for each access request. This results in a high setup cost but relatively low usage cost for SMART POLICIES. Finally, we remark that the block gas limit value in Figure 8.5 refers to our Academic testnet. Other blockchain could have a different gas limit, because the block gas limit can be initially set for each network.

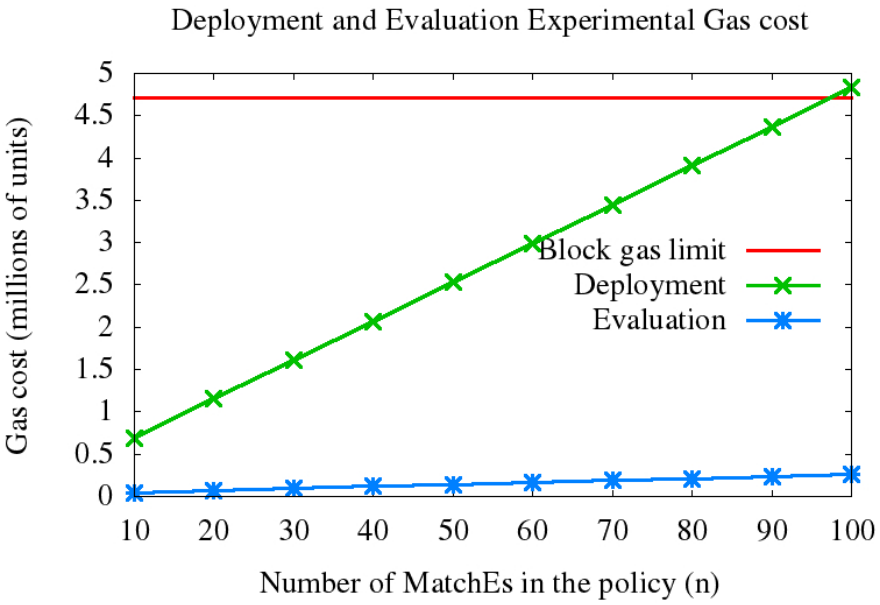


Figure 8.5: Gas cost of deployment and evaluation of the reference policy with increasing number of MATCHES n. The last value (n=100) is obtained by artificially increasing the block gas limit.

8.7.2 Resources

In our framework, the main need for computational resources is for running the blockchain client. Our reference implementation is based on the Ethereum client `geth`, that nowadays runs fine on standard hardware (i.e., two or more CPU cores, 4 or more GB of RAM memory, and a good network connection) but it requires some storage space (at the time of writing less than 200 GB for storing the main Ethereum blockchain using `geth`). However, since we delegate the storage of policies to the blockchain, we save the storage space required by traditional PAP implementations,

although this is not comparable to the space needed to save the entire blockchain. The storage space requirement can be an issue in some scenarios, to solve this issue two different solutions are possible. The first one relays on a third party that is in charge of managing the blockchain side of the client. Of course this introduces a new cost in the system as well as a point of centralization that needs to be trusted. The second solution, instead, is to use a light client to interact with the blockchain. This would result in a reduction of the storage requirements from hundreds of GB of memory to a few GB, at the expense of potential trust requirement (depending on how the light node is actually implemented) in other full nodes, i.e., nodes following the full protocol specifications and storing the entire blockchain. Currently `geth` provides a 'light node mode' but it is still in beta version. Choosing a light client based implementation would allow to deploy our system on most of the nowadays common machines. As already stated in Section 8.6, we remark that this is only an issue for the traditional digital resource reference example. In fact, in the reference example involving SMART RESOURCES, the user needs a blockchain client anyway to manage such SMART RESOURCES, and so our system could use the same client, not introducing additional resource requirements.

8.7.3 Time

The time overhead introduced by operations on the blockchain for the policy creation is caused by the SMART POLICY deployment phase, while at access request time, it is instead due to the execution of the SMART POLICY `evaluate` function. We do not consider the time needed for the user to check and sign a transaction since it is user dependent. Both the SMART POLICY deployment and execution times mainly consist in the time elapsed for the corresponding transactions to be mined into a block. So, they both are expected to be roughly equal, on average, to the transaction confirmation time, that depends on the underlying blockchain. In our case, the Ethereum blockchain is designed to add a new block on average every 14 seconds. This means that, as long as there is enough free space in new blocks, a new transaction should take, on average, 14 seconds. In practice, to estimate this cost, we should also take into account the transaction propagation times (that might be comparable to the mining time for poorly connected nodes). Moreover, in case of transaction congestion, a transaction can actually take longer to be included in a block. However, this time can be manipulated by our system, as well as by the other blockchain users, by choosing a more competitive gas price (i.e., choosing to pay more for faster confirmations). For these reasons, there is no simple way to determine how many blocks on average a new transaction will take to be confirmed. The most important factors influencing confirmation time of a transaction are highlighted more precisely in the following list:

- **Random mining time**: in a PoW based blockchain, blocks are generated at random times. It is guaranteed that the difference between the generation

times of two consecutive blocks is equal *on average* to a predefined constant (and PoW difficulty is adjusted accordingly). This means that, in practice, users can expect an high variance in the time to be waited between the generation of two subsequent blocks. This is especially true during difficulty adjustment periods. For example, if the network experiments a sudden drop or increase in the computing power of miners dedicated, this can negatively affect the block generation time (respectively increasing or reducing the expected average time) before the difficulty adjustment process can automatically keep up with the change. Also, during forks, the validation time might increase since the mining power is split between different branches, and each branch is working to solve a PoW with a difficulty not correctly proportionate to the actual computing power.

Supposing a constant flow of new transactions submitted to the network, the transactions that are submitted during the mining of the blocks that take more time, have to compete with more transactions waiting to be mined, and so will overall have a smaller probability to be added in that block with respect to the transactions that are submitted during the mining of the blocks that take less time. This is not a problem in general for a blockchain since the constant expected generation time guarantees that on average equal transactions will have the same chances of being added and so the same expected wait time to be validate (assuming blocks are not full). However, in case of time sensitive applications this could be an issue affecting user experience in some circumstances.

This issue can be mitigated using different distributed consensus algorithms with guaranteed constant mining times, e.g. Proof of Authority (PoA) [227], but such algorithms usually require to weaken the *lack of trust* assumptions of public blockchains and so are used for permissioned ones (see Section 1.6).

- **Discrete block creation**: a blockchain timestamps transactions by dividing the set of records in blocks. Basically, the system takes discrete temporal snapshots of its state at each block. This discrete time snapshots can be taken at constant times intervals on average (e.g., in PoW blockchain) or exact constant time intervals (e.g., in a PoA blockchain), as explained before. In both cases, transactions that are submitted closer to the moment when the next snapshot is taken have to wait less time for the next block and so a chance to be validated. In general, transactions submitted immediately after a new block is found have to wait more than transactions submitted immediately before a new block is found, assuming that there is enough free space in new blocks and miners keep listening and adding transactions to their blocks while mining. If blocks are saturated, instead, the opposite is true since equal transactions will have more chance to be added to the next block if they are submitted immediately after a new block is found.

- **Latency**: the chances for a transaction to be mined increase as more miners get to know about its existence. This means that transactions submitted by poorly connected nodes are expected to wait on average more time than identical ones submitted by better connected nodes. Furthermore, the unpredictability of network latency influences the time need for the transaction to be known and so its chances to be mined. Such latency is also influenced by the mean used to connect to the blockchain communication network. For example users connecting through a browser based service (such as the popular *MetaMask* [183]), where request needs to travel through the internet to reach a third party node directly connected to the communication network, will incur in an higher latency, in general, then users connected directly to the network through a full node. Finally latency can also be manipulated by malicious entities trying to isolate nodes to delay their transactions (for example during double spending attempts).

- **Congestion**: as block space gets exhausted it increases the competition among transactions. The same transaction will have to wait more for validation during times of high congestion with respect to times that free space in blocks is more abundant. Do note that in this paper we consider as *block congestion* the ratio between the total gas used by all transactions in a block and that block gas limit.

- **Fees**: as already noted above, users can decide to offer higher fees for their transactions to be more desirable by miners. This becomes more relevant during high congestion periods. In fact, if space is available in blocks than it is convenient for miners to insert any transaction as long as the fees repays for their effort (e.g., computing the contract contained), so users have no reason to offer higher fees. On the contrary, if blocks are full (i.e., there are more pending transactions than the space in a block to contain all of them) then the miners will choose only the higher paying transactions to reap higher rewards with the same effort (since the *gasCost* is independent from the gas actually consumed).

- **Lack of a global clock**: as for most distributed systems, the communication network lacks a global clock. This means that each node, and, especially, each miner has its own local time that may differ (even significantly) from the ones of others. Since miners solving a block are the one deciding the block timestamp, this means that blocks can have inconsistent times among them. Currently, Ethereum only requires that a block timestamp must be greater than the previous block timestamp [276]. So miners can maliciously alter their own timestamps. To mitigate this problem, each node could use its local timestamp to keep a local log recording when transactions and blocks are first seen by such node. However, this is not a good solution as well, since, as already explained, network latency could cause a transaction or a block to

be seen by a (poorly connected) node after a significant delay with respect to the time when, respectively, the transaction has been submitted or the block has been created.

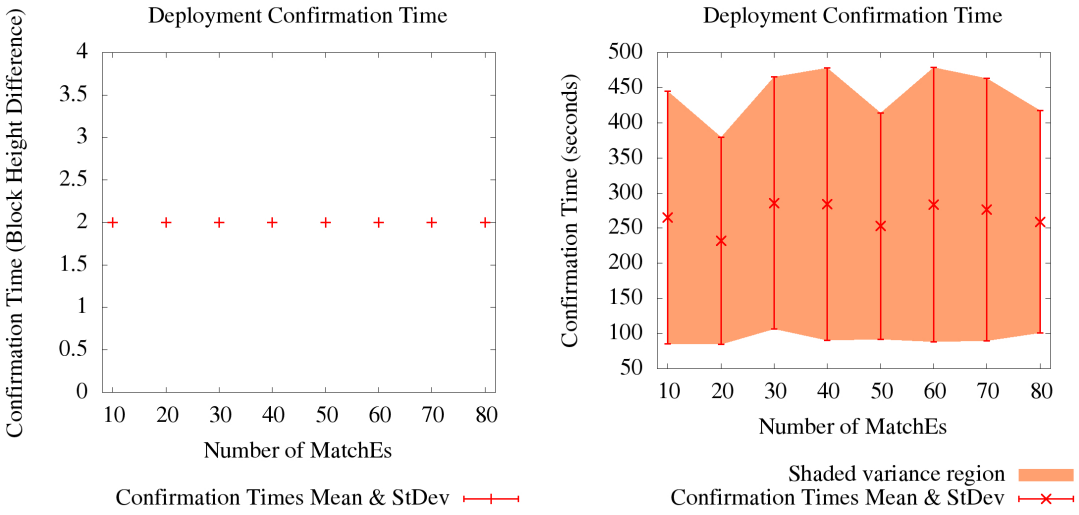
- **Context dependent evaluation** (evaluation only): the same transaction containing a function call can take different times to compute depending on the current context it is executed in. This is strictly connected to the monetary cost we explained before in Section 8.7.1, here it suffices to say that since a function execution depends on the current state it can have different costs and so require more space in a block. This influences its probability of being mined sooner. For example we can think of a function that has a conditional clause depending on the current block height number, if it is even, it terminates, otherwise it does a lot of heavy computations. Clearly the function will require more computations and so it will cost a lot more gas if the current block number is an odd number, and heavier transactions have longer expected validation times.

To alleviate all those issues and reduce the variables that concur to determine a transaction confirmation time, it is generally advisable to consider the blockchain itself as measurement of time through the *block height* (i.e., the distance of a block from the genesis block). In fact, by grouping the transactions in immutable discrete snapshots (blocks), it is possible to see the blockchain as a timestamping service. Using such a solution the confirmation timestamp of a transaction is the height of the block it was mined in. This means that we can measure the time it took to confirm a transaction as the difference in block height between the last known block that the network had mined at transaction deployment time (relative to the transaction creator) and the block where the transaction has been inserted. This solution solves the *random mining time*, *discrete block creation* and *lack of global time* problems showed above. Considering that *latency* is an unsolvable problem inherent of the system and the *context dependent evaluation* problem can be avoided by choosing functions and context appropriately, using the block height allows us to perform experiments with only *congestion* and *fees* variables to take into account. This way, even if we still have some degree of unpredictability due to latency, it is easier to isolate the cause of long waits and delays.

Using block heights as timestamps for transaction confirmations is especially effective in our academic testnet. In fact the network is heavily underused and so there happens to be long periods of time during which the blocks are almost empty. This allows us to perform experiments without worrying about block congestion and fee races.

To test the deployment time of a SMART POLICY we measured the confirmation time, expressed in block height, of a reference SMART POLICY, increasing the number of MATCHES, used as an approximated measure to estimate policy complexity. Each measurement has been repeated for 50 times. As reference policy we used the aforementioned reference policy showed at the end of Section 8.7.1, i.e., a SMART

POLICY invoking a single SMART AM and built by a conjunction of n simple single attribute MATCHES crafted to be always true over the SMART AM returned values. Furthermore, to avoid to introduce artificial congestion in the blocks with the experimental data themselves, we also temporized the deployment of each policy by waiting for one block after each successive contract deployment.



(a) (b)

Figure 8.6: Average (with standard deviation) deployment confirmation times, expressed through block height difference (a) and in seconds passed between the current block timestamp and the timestamp of the block the transaction gets confirmed in (b). Do note that in (a) the standard deviation is always zero. Experiments on the Academic testnet.

Figure 8.6(a) shows the average deployment confirmation time (expressed in number of blocks) and its variance for an increasing policy complexity, expressed by the number of MATCHES in the SMART POLICY. We do note that 1 is theoretically the optimum time, because it means that the transaction is confirmed in the first block available. We observed that, in our testnet, the actual optimum is 2 instead of 1, i.e., no transaction is inserted in the immediately available block. This fact can be explained as consequence of the mining behaviour of our nodes. In fact, it looks like each node, once has created a candidate block and started to try to solve the corresponding proof of work, ignores new transactions until it succeeds or some other nodes solves its proof of work first. This means that new transactions are considered for insertion in new blocks only after the first block that could contain them is mined, and so they have to wait at least two blocks to be confirmed. The situation is different for the Ethereum main chain which which is currently experiencing a more relevant congestion problem, as we show in Figure 8.7.

Figure 8.6(b) shows the results of the same set of experiments considering block confirmation time measured in seconds instead of using block height. Hence, the

transaction confirmation time was measured as the difference in seconds between the timestamp of the transaction confirmation block and the timestamp of the last block known by the transaction creator at the time the transaction was broadcast to the network. Moreover we point out that the block confirmation time in our academic testnet (that has been set up to 136 seconds at the time of our experiments) is higher with respect to the main Ethereum network (14 seconds). It is clear from the comparison of the two figures how using block height as time measure gives a cleaner understanding of the phenomenon. In fact the comparison between Figure 8.6(a) and Figure 8.6(b) shows the impact of the *random mining time* issue. We can see clearly in Figure 8.6(a) that the number of MatchEs in the smart policy has no effect on the deployment time as long as blocks have enough free space to include the policy deployment transaction. Instead, from Figure 8.6(b) it looks like smart policies with 20 or 50 MatchEs have shorter deployment time. Of course there is no reason why this should be the case, as proved by Figure 8.6(a) that is derived from the same set of experiments, it is just a consequence of the randomness of mining times manifested by a relatively low number of experiments (50). We can compute a theoretical expected validation time as two times the block confirmation time, i.e., $2 * 136 = 272$ seconds (although an unlucky combination of all the causes affecting confirmation time listed before could cause an extraordinary long wait). We do note that the expected validation time matches with our experimental results, as the average of all confirmation times measured is 267.34 seconds, close to the theoretical value of 272 seconds.

As shown in the previous paragraph, performing experiments on our academic testnet allowed us to prove some theoretical assumptions, but was not a good simulation of a real world scenario. To have performance evaluations closer to a real scenario, we decided to perform our experiments also on the public Ethereum testnet Ropsten. Considering that it is widely used globally by many users, testing their applications before deployment on the main Ethereum chain, we believed this was a better simulation of a real world scenario. In fact both the expected mining times (14 seconds) and block congestion of Ropsten are closer to the ones of the Ethereum main network. So we repeated the same experiments on the Ropsten testnet, repeating each measurements 10 times for each experiment. The deployment confirmation times results, measured in block height difference, are shown in Figure 8.8(a).

Figure 8.8(a) shows that the confirmation times of the experiments concerning the smart policies with 80 MatchEs are considerably larger w.r.t. the experiments concerning smart policies with a smaller number of MatchEs. Moreover, this problem has not been detected in our academic testnet, as shown by Figure 8.6(a). This happened because the size of a transaction to deploy a smart policy with 80 MatchEs is close to the *gaslimit* of the blocks (see Section 8.7.1), so this transaction needs blocks with a lot of free space to be confirmed. This is not an issue in our academic tesnet, where most of the blocks are empty. Instead, finding empty blocks to fit this transaction is more difficult in a real world network

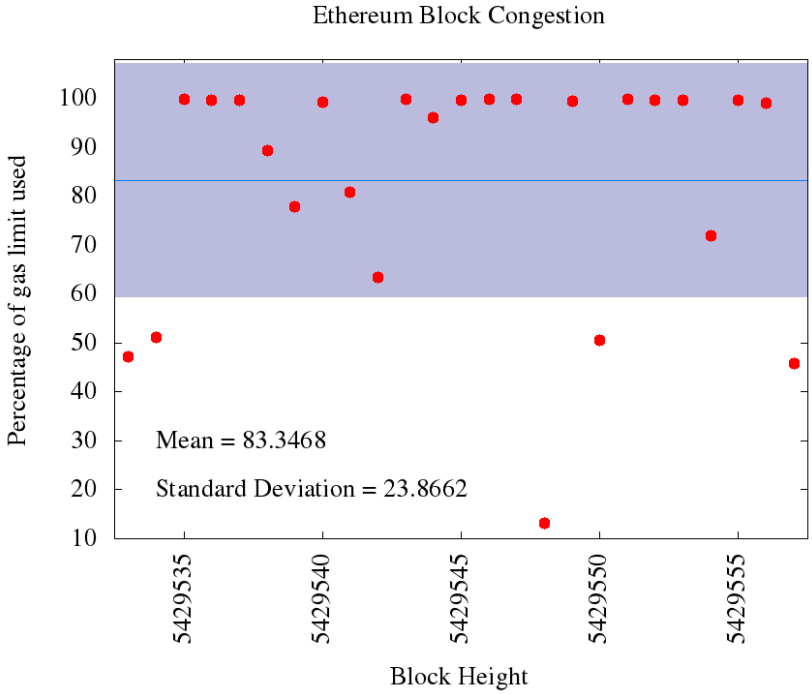


Figure 8.7: Percentage of gas used in 25 consecutive blocks of the Ethereum main chain from block 5 429 533 to block 5 429 557 [102], average congestion 83.35% with standard deviation 23.87 .

such as Ropsten. In Figure 8.8(b) we show the same results zooming on the first seven sets of experiments only to give a better insight excluding the 80 MatchEs special case.

The results are worse than the ones obtained for the academic testnet, and shown in Figure 8.6(a). Of course this was expected since Ropsten is more congested than our academic testnet. We note that in Ropsten is possible to achieve the optimum result of mining a transaction in 1 block only, and this is due to a smarter mining policy adopted by the Ropsten testnet nodes. We also note that we obtained an average confirmation time quite low (always less than 4 blocks), and independent from the number of MatchEs. Comparing this with the results from the experiments with 80 MatchEs, we can conclude that the number of rules in a SMART POLICY does not influence confirmation times as long as the resulting transaction is small enough to fit into the average space available in blocks. To further prove our point we measured the average block congestion during the experiments we conducted. The results are depicted in Figure 8.9. To measure the average block congestion experienced by a transaction, we compute the average of the congestion values of each block that the transaction had to wait for its confirmation. We then compute the average among the 10 measurements performed for each experiment. We can see that block congestion does not influence the block confirmation times. The case with

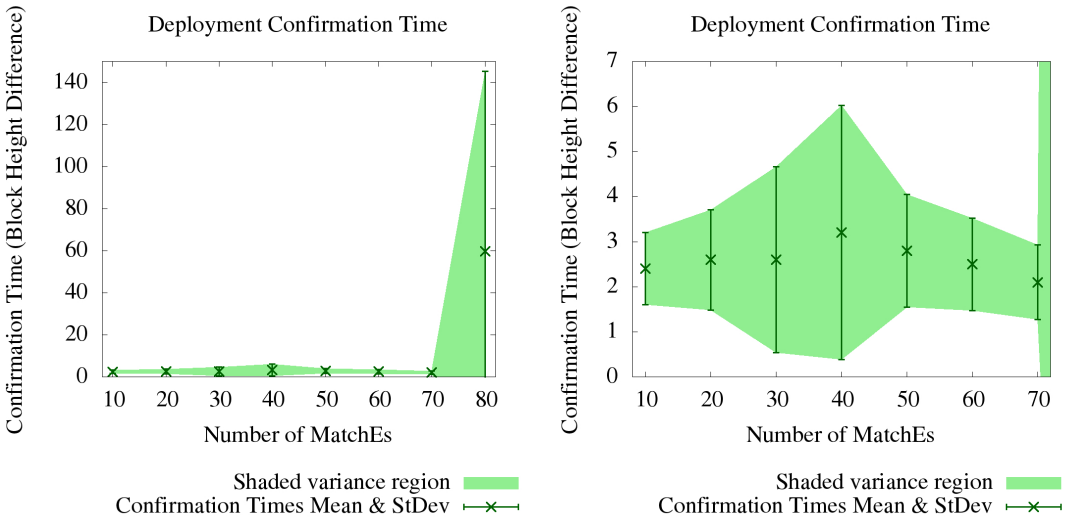


Figure 8.8: Average (with standard deviation) deployment confirmation times results expressed through block height difference. Complete (a) and excluding the last set of results with 80 MatchEs (b). Experiments on the Ropsten testnet.

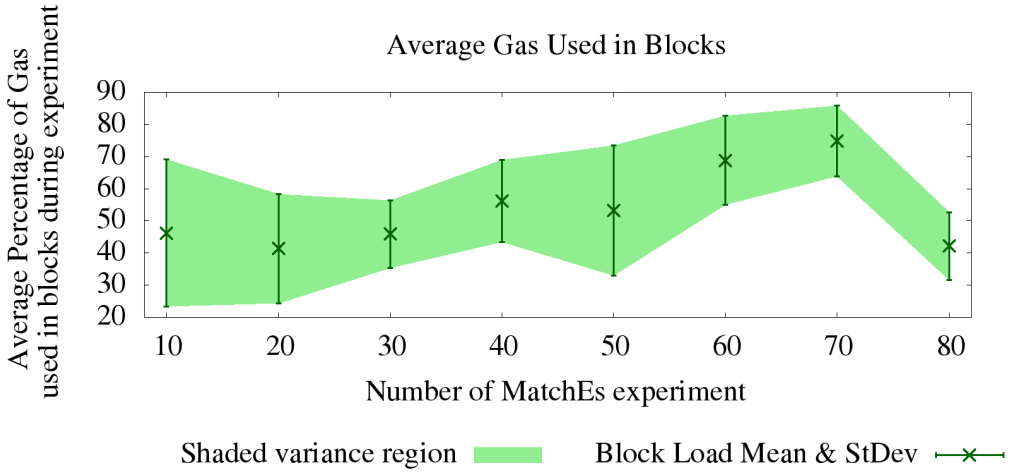


Figure 8.9: Average block congestion times experienced during the deployment transaction confirmation experiment shown in Figure 8.8.

highest block congestion (i.e., 70 MatchEs) is also the one with a lower average confirmation time.

Also for the Ropsten testnet we measured the time in seconds instead of considering the block height difference. The corresponding results are shown in Figure 8.10. The same considerations made before for Figure 8.6(b) do hold also in this case. We observe that even if the transaction confirmation times expressed in block height

difference are higher in Ropsten than in the academic testnet (see Figure 8.8(b)), thanks to the much lower average block confirmation time (14 seconds vs 136 seconds), the resulting average confirmation times in seconds are lower in Ropsten (always below 50 seconds) than in the academic testnet.

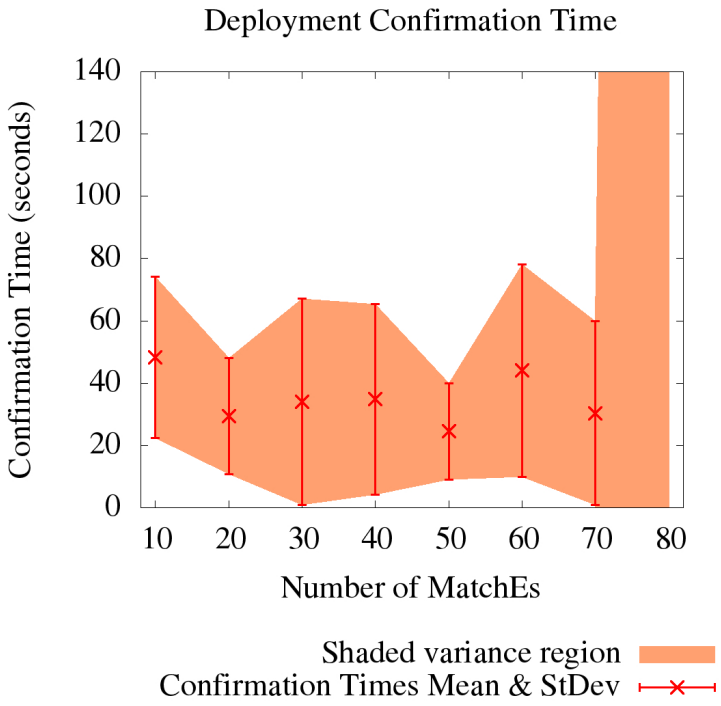


Figure 8.10: Average (with standard deviation) deployment confirmation times results expressed in seconds passed from the current block timestamp and the timestamp of the block the transaction gets confirmed in. Experiments on the Ropsten testnet. The plot does not show the last set of experiments for 80 MatchEs for clarity reasons, the missing values are *Mean 819.6* and *Standard Deviation 1219.5* .

As we have shown in Section 8.7.1 the SMART POLICY deployment transactions need to store a new smart contract on the blockchain and execute its constructor. This often results in transactions consuming a considerable amount of gas and space in a block. In comparison, SMART POLICY evaluation transactions (i.e., transactions executing calls to the SMART POLICY `evaluate` function) are in general much lighter. This causes those transactions to be easier to be added to a block and so they take, on average, less time then deployment ones. This is a good property since each SMART POLICY needs to be deployed only once, but can be executed several times (as long as the policy remains active). We decided to give a measure of the system *time efficiency* by studying the execution of SMART POLICY evaluation transactions alone. As time efficiency measure we chose to use the *execution throughput*. Given

n SMART POLICY evaluation requests submitted at the same time, we define as *execution throughput* the percentage of them that got evaluated immediately (i.e., got confirmed in the first available block). Instead the *execution time* is simply the average time that took for all the n requests to be evaluated. As explained above, the *context dependent evaluation* issue may affect these results. To mitigate this, we chose to execute all our experiments on the reference policy described above, which is built to be non short circuiting, thus ensuring that all the MATCHES of such policy are always all evaluated in our tests. Furthermore we performed our experiments on the Ropsten testnet since it better reflects a real world scenario as explained before. We measured both execution throughput and execution time for an increasing number of requests and of MATCHES in the SMART POLICIES. In particular, we performed a set of five experiments for each fixed number of evaluation requests, the number of simultaneous evaluation requests were increasing from 10 to 30, then 50, 70 and finally 90. These experiments were repeated calling the evaluation of a reference SMART POLICY with 20, 40, 60 and 80 MATCHES. We chose to repeat our experiments only five times for each setup and only for four different SMART POLICIES to avoid excessive network bloating. In fact, the presented experiments already involved five thousand single transactions. Since we performed those experiments on the global official testnet we tried to keep the number of transactions contained to avoid affecting all the other users concurrent tests. The cumulative results showing the average *execution times* expressed in block height difference are reported in Figure 8.11, while in Figure 8.12(a), Figure 8.12(b), Figure 8.12(c) and Figure 8.12(d) are shown the individual results for each SMART POLICY. As expected we can notice from the figures how the average confirmation time increases with the number of evaluation requests. This is expected since more evaluation transactions compete for the same finite space in each block.

In Figure 8.13(a), Figure 8.13(b), Figure 8.13(c) and Figure 8.13(d) are shown the results of the same experiments of Figure 8.11, but considering the *execution throughput* instead of the *execution time*. The definition of *execution throughput* is relaxed to consider the percentage of transactions that are confirmed within x blocks, with x a parameter. The figures show the results for x between one and four. Unsurprisingly and despite some exceptions, probably caused by random block congestion, the results show the expected trend that more requests result in a lower *execution throughput* (as already verified for *execution time*). We do note that no transaction is ever confirmed in one block (i.e., inserted in the first available block), this is an artificial consequence of our experiments manager program. In fact, we first observe the current block (i.e. the last block known by our node) and then spend some initial time to create evaluation requests and broadcast all transactions at once. This, coupled with network latency, may be an explanation for the observed delay, that, for example, was not present during the deployment time experiments (where many SMART POLICIES got deployed in just one block, see Figure 8.8(b)).

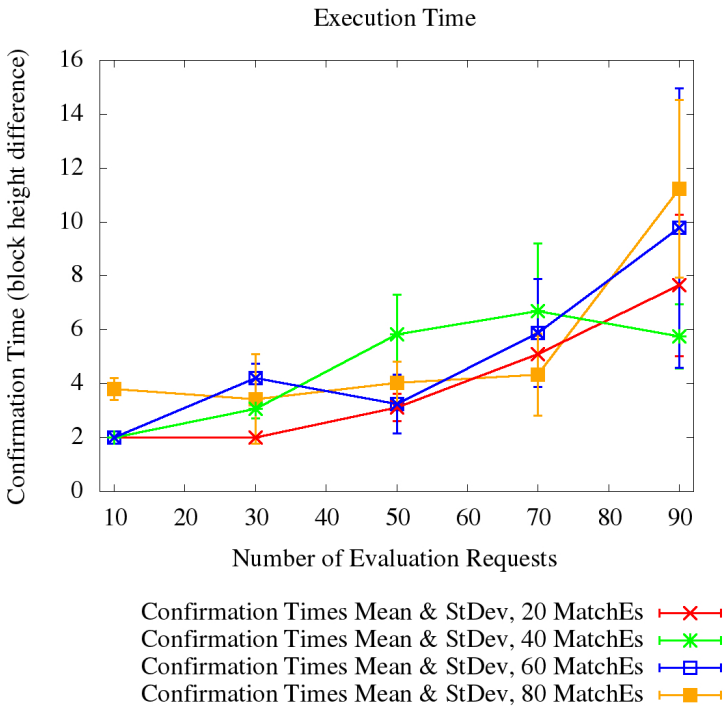


Figure 8.11: Average (with standard deviation) evaluation confirmation times results expressed in block height difference, for increasing number of execution requests and number of MATCHES in the reference SMART POLICY executed. Experiments on the Ropsten testnet.

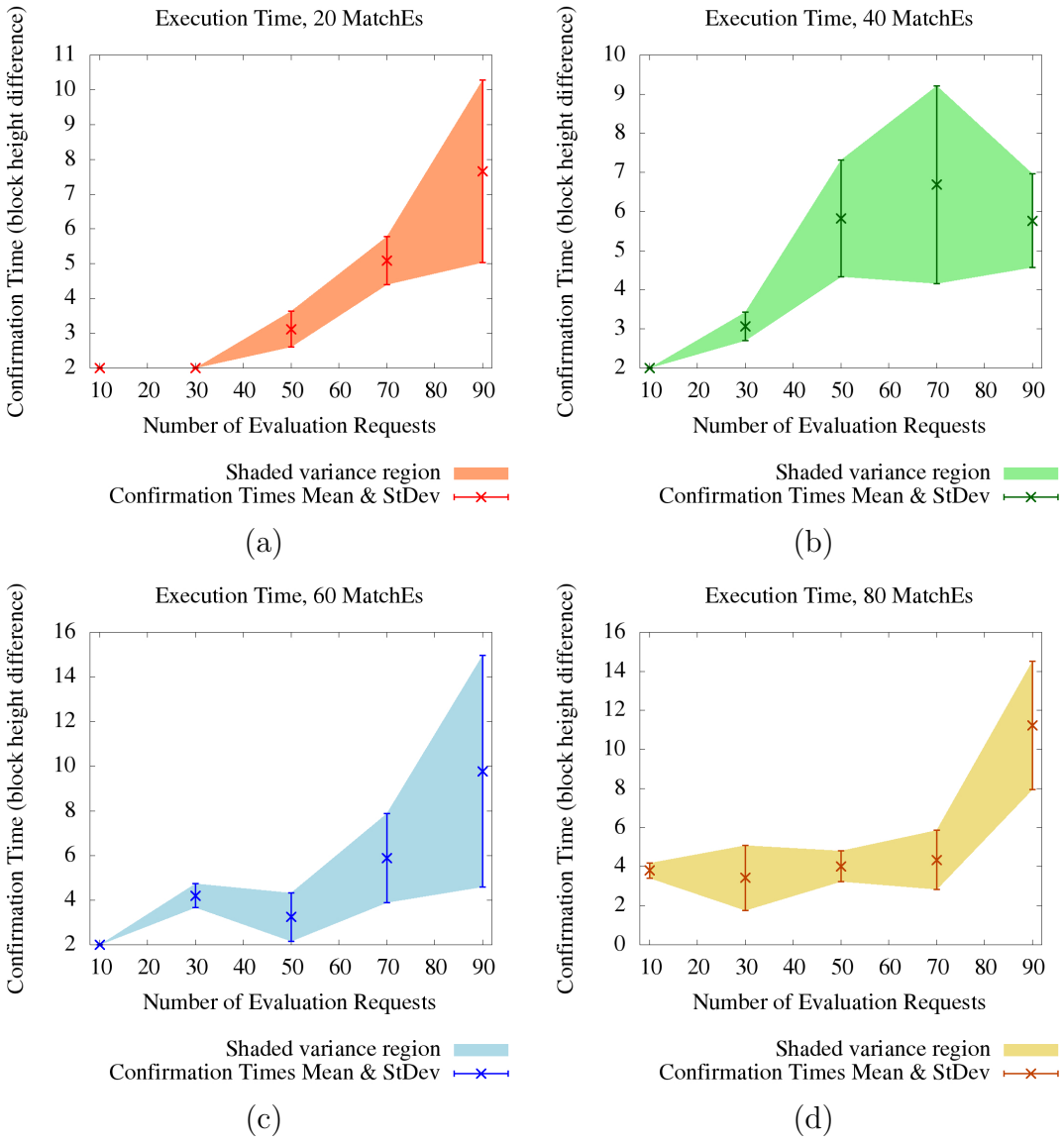


Figure 8.12: Average (with standard deviation) evaluation confirmation times results expressed in block height difference, for increasing number of execution requests to a reference SMART POLICY with 20 (a), 40 (b), 60 (c) or 80 (d) MATCHES. Experiments on the Ropsten testnet.

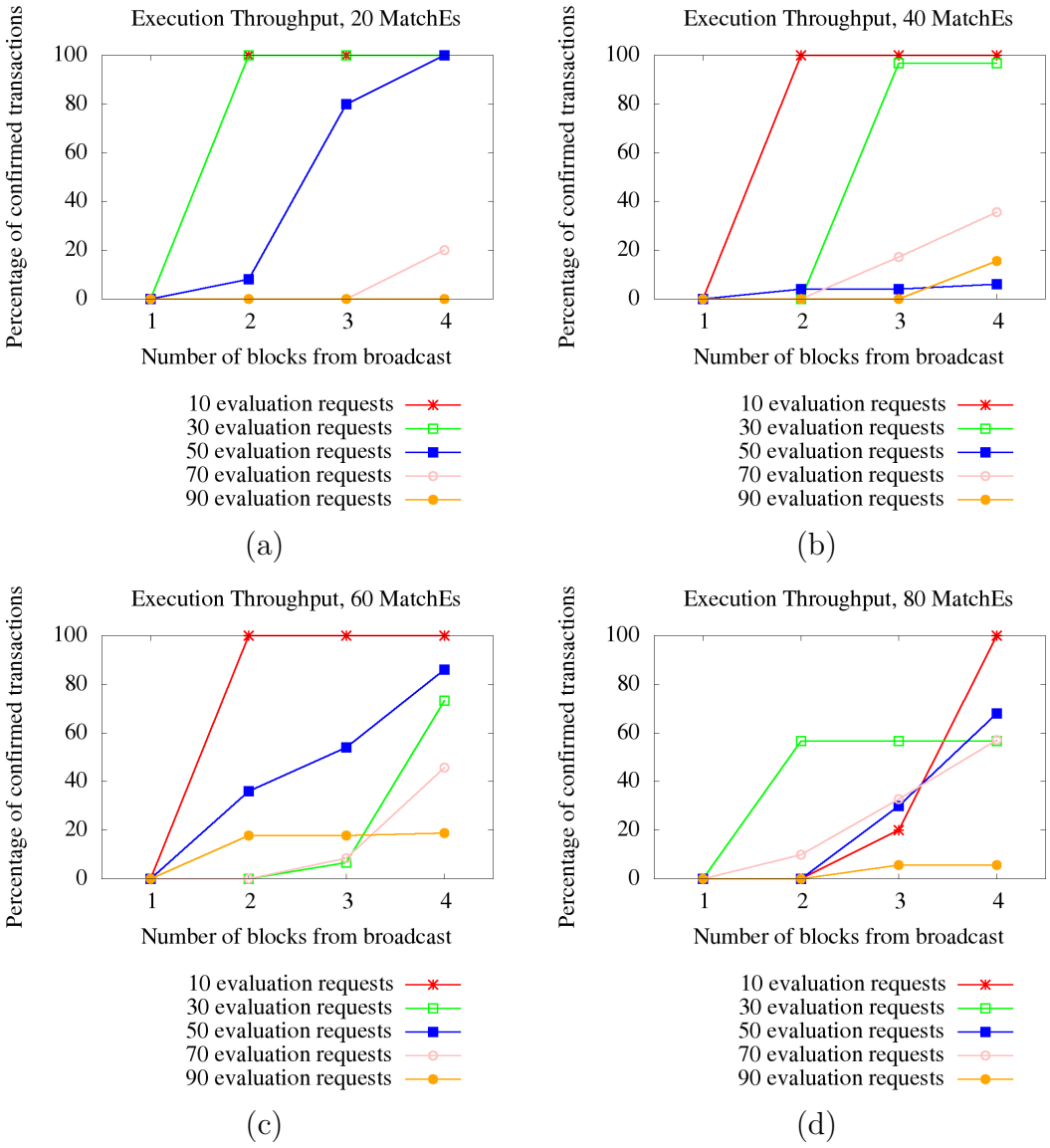


Figure 8.13: Average percentage of confirmed transactions in x blocks, for increasing number of x and increasing number of requests. Experiments regarding a reference SMART POLICY with 20 (a), 40 (b), 60 (c) or 80 (d) MATCHES on the Ropsten testnet.

9

Conclusions and Future Work

Abstract

In this final chapter of Part II we present our conclusions about our proposals of blockchain technology integration with Access Control systems. After showing our current research directions, we introduce a possible extension to the blockchain based Access Control service showed in the previous chapter. The proposal shows a minimal use of blockchain technology in a traditional Access Control system, by only implementing Attribute Managers as smart contracts.

In Part II we have presented our blockchain application proposal related to Access Control Systems. Our main contribution is the integration of blockchain technology with Access Control systems to obtain a blockchain based Access Control service. Such novel system inherits both advantages and drawbacks of blockchain technology.

In our opinion the main advantage obtained is auditability. In fact, delegating the Access Control policies management and evaluation (through smart contracts) to the blockchain delegates such tasks to a decentralized, transparent, and immutable system. This means that resource owners can not fraudulently deny access to subjects without leaving an auditable trace of the misbehaviour.

Instead, the main disadvantages are performance and scalability issues of the underlying blockchain. Due to its decentralized nature, using a blockchain inevitably introduces different time and cost constraints with respect to any centralized solution. A deeper analysis of pros and cons is provided in Section 8.6.

Our main future work direction of research is split among different topics.

- First we plan to properly address the possible privacy issues of our proposal. In fact, in order to provide complete auditability, our system stores all information publicly visible on the blockchain (either as simple data or stored inside the state of the smart contracts involved). It is worth studying which is the best method to mask such data while still allowing auditability, if necessary weakening it by providing it only to the parties involved.

- Another interesting research direction is expanding our work to encompass more Access Control standards and models. For example, we are studying the

possibility to extend our approach in order to support more complex policy sets or to deal with advanced authorization models, such as the Usage Control one.

- Finally we are studying better ways to implement an Access Control service, analyzing possible different mappings of the XACML reference architecture logical modules to blockchain and off-chain functionalities.

Further extensions of our proposed model are also possible while still maintaining its core components. In Chapter 8 we have shown two different reference examples. The first represented an hybrid approach, where some of the components were still managed in a traditional off-chain fashion, while the rest was deployed on chain (see Figure 8.3). In the second example instead we deployed on the blockchain all possible components, leaving only the inevitable ones off-chain (see Figure 8.4). An opposite approach could be to try to maintain a traditional Access Control system and deploy on the blockchain only the components that are already externalized in such systems. For example, in current Access Control systems the Attribute Managers are already managed by external third party services. This means that these components could be deployed on a blockchain as SMART AMs, as explained in Section 8.2.2. The preliminary results of this work are presented in the next section.

9.1 Attribute Managers Only on the Blockchain

We have shown in Section 8.4 our proposed hybrid system to deploy and evaluate XACML policies based on blockchain technology. We have expanded this approach in Section 8.5 to show how our system could be embedded even further in a blockchain to provide a classical XACML Access Control system applied to blockchain assets (i.e. smart contract functions). In this section we consider the opposite case, i.e. how a traditional XACML Access Control system may interact with the blockchain without changing its core functionalities.

In the traditional XACML system there are already entities that are external to the system: the AMs. The attribute providers are usually third party services that are queried by the local PIPs to obtain the correct attribute values at request time. As we have already shown in Section 8.2.2, the tasks of a traditional AM can be easily coded into a smart contract on the blockchain (i.e. SMART AM), so we can imagine a scenario where some AMs are actually deployed as services on a blockchain. In this section we present how a traditional XACML system can be adapted to work in such a scenario.

In other words, we have presented in Section 8.4 how an hybrid system could be built to rely on a blockchain to execute part of the Access Control tasks (see Figure 8.3). In Section 8.5 we have shown a fully integrated blockchain system, where all the possible access tasks have been delegated to the blockchain (see Figure 8.4).

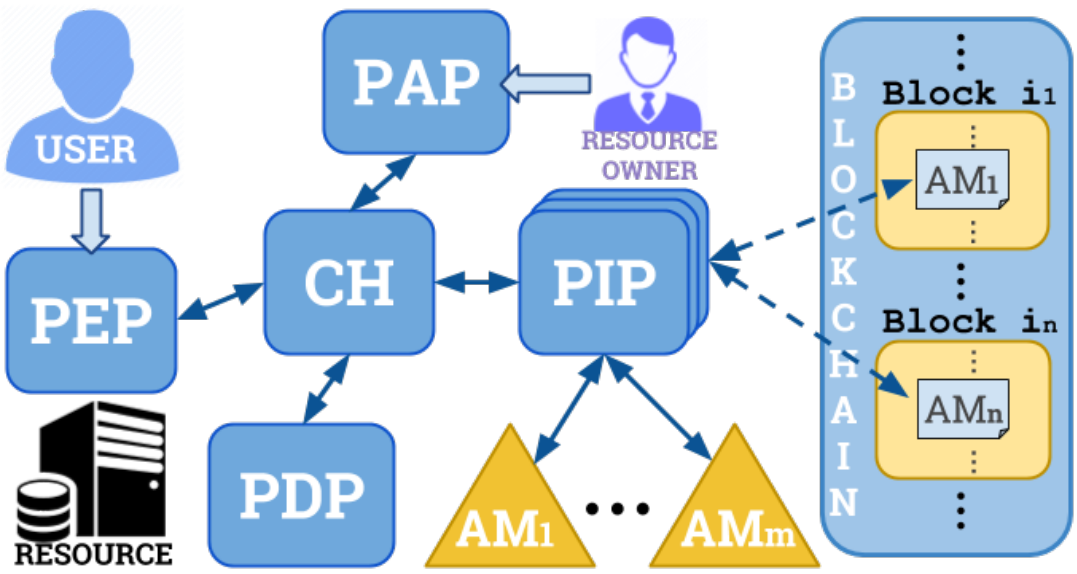


Figure 9.1: Architecture of AM only blockchain based Access Control service.

In this section we propose a system where only the minimal amount of Access Control tasks possible are blockchain based (see Figure 9.1).

The advantage of using SMART AMs instead of traditional AMs is in the added properties provided by blockchain technology. Since a SMART AM is a smart contract it maintains all properties they have (see Section 1.6). For example it is more secure than a centralised service because it removes the problem of single point of failure and attack. Being executed on the blockchain, it also guarantees to be always available (as long as the underlying blockchain is available). Furthermore in a public scenario it also allows increased transparency since the returned attribute values cannot be tampered with by the AM owner (unless it is publicly stated by the contract code).

PIPs are already pluggable components in a traditional XACML architecture. Since they need to know the format of queries to retrieve attributes from different AMs they can be custom built with a singe AM in mind. This means that to adapt the traditional system to support SMART AMs we only need to define a new pluggable PIP capable of interacting with a smart contract.

Following our reference implementation scenario (see Section 8.3.1) we created a proof of concept implementation of a PIP written in java and capable of interacting with SMART AMs deployed on the Ethereum blockchain. As explained in Section 8.2.2 we imagine an already existent ecosystem of SMART AMs advertised by their respective owners. We assume the owner lets the potential users know at least the contract address and the mean to retrieve attribute values from it. For example it could advertise the signature of its public methods returning values for certain attributes, or the name of the local variables in the contract state to be read

to retrieve a value. The advantage of this second solution is that it involves non contract code execution to be performed and so can be done free of gas expenses. Of course it is in the interest of the owner to make those information reachable to potential users (customers in case of pay to use SMART AMs). Assuming that our PIP wants to retrieve the current value of the attribute a from the SMART AM am it only needs to execute the corresponding function of am or read the corresponding value in am to obtain the up to date value for a. This means sending a transaction to the blockchain or simply reading a value from its state. Our PIP is implemented in java as an external component pluggable to a traditional Access Control system. This PIP automatically builds the correct request (either a transaction containing a SMART AM function call or a SMART AM state read) depending on the attribute name and using its internal tables built during creation time based on the SMART AM advertised information. It then uses `web3j` to send the request to a `geth` client that will use it again to send back the answer to the PIP.

A

BITKER: a P2P Kernel Client for Bitcoin

In this Appendix chapter we present BitKer, a kernel client which can interact with the Bitcoin Peer-to-Peer (P2P) communication network. The proposed client provides an API which can be exploited by applications requiring to interact with the Bitcoin P2P network. The client has been exploited to perform three different logs of the network. The logged information has enabled the classification of the Bitcoin clients active in the network, the study of their average connection time and number of messages exchanged. Results show that most part of Bitcoin clients are official. An interesting finding is the sudden change of the type of Bitcoin clients after the August 2017 *Bitcoin Cash* hard fork.

Overview. Our listener is implemented as a Bitcoin *kernel client*, i.e. it manages all basic network related operations of a Bitcoin full client, with the possibility to plug in more advanced applications on top of it. In fact the aim of the project was to develop the lighter possible Bitcoin client, still behaving in the expected fashion of a fully fleshed out full node. We can think of our client as a gateway for more complex applications to interact with the Bitcoin communication network, either by listening and logging specific information (e.g. all transactions of a certain type, or all transactions from a certain node or IP range) or by sending transactions out. The main goals desired from the client were for it to be always active and connected to as many nodes as possible. Satisfying the first requirement meant implementing a series of mechanisms to periodically save the client state to quickly recover from failures in a best effort way, i.e. trying to revive the active connections the clients had before the crash. Instead, keeping an high number of connections (in practice we experienced around four thousands active connections at any time) required special care in the project design to allow it to sustain the heavy load of messages incoming and directed to thousands of nodes at the same time. To increase the number of active connections our client was periodically trying to connect to known nodes and constantly requesting information about their know nodes lists from its neighbours.

To keep our client as light as possible we decided to implement with it only the network related functionalities of a Bitcoin full a client. In fact, our client was compliant with the protocol recommendations only for the rules necessary not to be

banned by other nodes. For example our node never requests data as answer from an *inventory* message, pretending, instead, to know the data already, unless, of course, the user specifies otherwise. This design choice was made because data transmission messages are the heaviest in terms of bandwidth consumption (especially if the data represent a block), and so could have bloated our node connection, increasing the latency of its messages. This implies, for example, that our client is not actually storing and updating the blockchain, and so it is can not perform validation tasks about received transactions or blocks.

This Appendix is structured as follows: Section A.1 describes the P2P communication network of the Bitcoin protocol. Section A.2 describes BitKer, in particular the overall structure of the client. In Section A.3 we introduce the communication protocol used by BitKer by listing the messages treated. The experimental evaluations are presented in Section A.4, and finally Section A.5 concludes the Appendix and presents future research directions.

A.1 The Bitcoin P2P network

We refer to the Satoshi client[1] for the definition of the P2P Bitcoin protocol. The protocol supports the bootstrap of the nodes on the Bitcoin network and the creation of a public global log of the state of the system. This public global log is the blockchain where transactions are stored.

The Bitcoin network is an unstructured P2P network [257], where the discovering of new nodes is done in three different ways:

- through hard coded DNS servers which provide a list of IP addresses of active nodes;
- through a gossip protocol which supports the exchange of addresses of nodes between neighbours;
- through a set of hard coded IP addresses which can be used when DNS servers do not work.

The communication protocol is a gossip protocol [152] implementing peer discovering and blocks/transactions propagation. The network discovery is implemented through *ADDR* and *GETADDR* messages. In the bootstrap phase, a node joins the network by receiving a set of peers with one of the three methods shown before, and then it obtains further addresses by contacting these peers. By referring to the Satoshi client, each peer initiates a connection to up to 8 peers, and maintains a maximum of 125 total connections (rejecting any other connection request). Peers can request addresses by using the *GETADDR* message and advertise known addresses by using *ADDR* messages. Each peer maintains a database, called the

[1] `https://github.com/bitcoin/bitcoin`

addrMan which contains a list of known addresses of other peers, and each address has a timestamp which is used to determine its freshness. The addresses in the database are exchanged with other peers in the network.

A gossip protocol is also exploited to propagate blocks and transactions in the P2P network. When a node in the network receives a new transaction (or block), it sends an advertisement message, i.e. an *INV* message, to all its neighbours which contains the hash of the announced information. Then, the neighbours can require the information that they need.

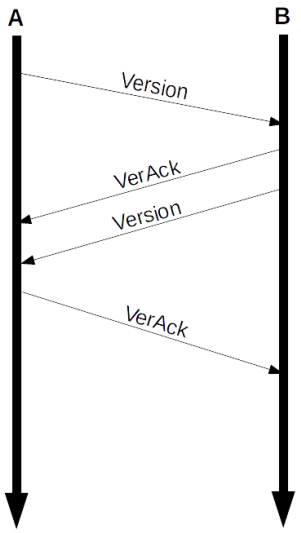


Figure A.1: How the version message is used: the node A has started a connection with B and it sends the version message, B, in turn, sends the same message. After this phase the two nodes will start to communicate with the same protocol

A.1.1 Message Structure

Each message exchanged in the network is divided into two parts: a header of 24 bytes and a payload with a variable length. Numbers are coded in little endian except for the port numbers and the IP addresses. The header is divided into 4 fields:

- Magic number, 4 bytes, which indicates the network from whom the message is arrived[2]
- type of the message (12 bytes);
- payload length (4 bytes);

[2]The Bitcoin network is divided into 4 sub-networks: the real network, two testnet, and the namecoin network which uses the blockchain as a database of couples (key,value)

- checksum (4 bytes).

A.1.2 Message Types

We describe the most important Bitcoin message types that are used to initiate connections and propagate data.

The VERSION Message This message is the first message sent after a connection has been established. In Bitcoin, blocks are transmitted by performing a header synchronization [124], and the version message is important because it is used to implement the handshake with the remote node. Each version message contains the IP and the port of the sender, the version number, a user agent string to identify the Bitcoin client the node is using, the number of the highest block it currently has, and a timestamp. As shown in Figure A.1, when the node A wants to connect to the node B, it sends to B a *VERSION* message. If the node B wishes to accept the connection, it acknowledges the message with a *VERACK* message, and it replies with its own *VERSION* message. Finally, if A wishes to proceed with the connection, it sends its *VERACK* and the communication can start.

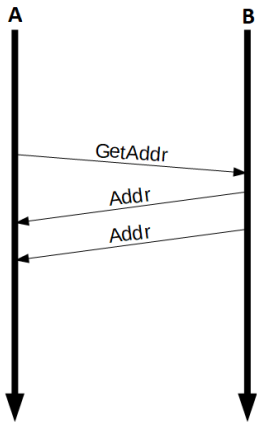


Figure A.2: Messages for Network Discovery

The GETADDR and ADDR messages The *GETADDR* message is used to send a request to a node asking for information about known active peers. The message does not have a payload. The list of addresses obtained thanks to this message is not the complete list of addresses a client knows. The *ADDR* message is used to relay connection information for peers on the network. The message is sent to a node which wants establish new connections and it contains the IP addresses

and ports of other peers. Most part of the received addresses can be addresses of nodes which are no more active, because they are obtained in the bootstrap phase or through other *ADDR* peers.

The INV message The *INV* message enables a peer to announce a block or a transaction by publishing only its hash. This allows to optimize the bandwidth, since transactions and blocks are sent only when a peer really needs them. As described in Figure A.3, when the node A creates or receives a new block (or transaction), it announces it to its neighbours by using an *INV* message, containing the type of data (transaction/block) and its hash. When a node B receives the *INV* message sent by A, it can request the data through a *GETDATA* message. A transaction is sent by a *TX* message while a block through a *BLOCK* message.

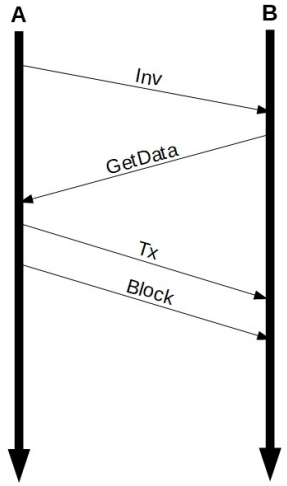


Figure A.3: Announcing transactions and blocks

A.2 BitKer: an overview

BitKer is a Bitcoin Kernel client, written in Java, which is able to connect to the Bitcoin network to establish connections to as many nodes as possible, remaining always active. The interface of the client is minimal, but it provides an essential API that can be used to interact with the network. One of the main goal of the client is to be reactive and efficient.

The structure of BitKer is shown in Figure A.4. The two main components are the External Interface and the Kernel Process. The External Interface is useful to provide live information about gathered data, instead the Kernel process is responsible of the interaction with the Bitcoin network. In the following we describe more in detail these components.

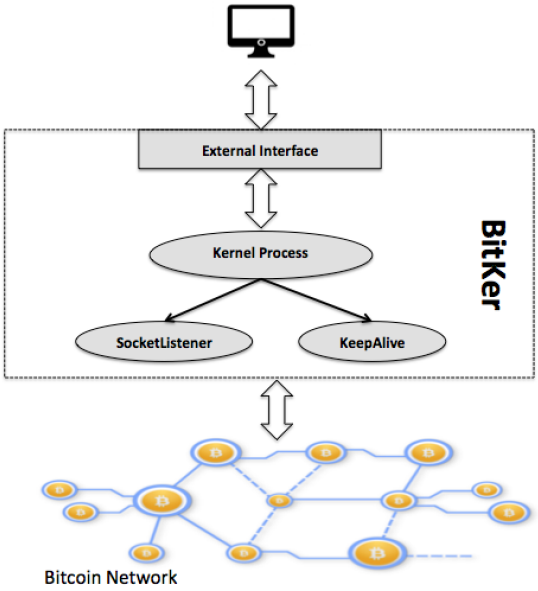


Figure A.4: The architecture of BitKer

A.2.1 The Kernel Process

The Kernel process interacts with the P2P network to obtain live information about the nodes present in the Bitcoin network. Every minute, the process chooses 1250 IP addresses, from the list of addresses obtained by the already connected peers, and it tries to connect to all of them.

The choice of IP addresses takes into account the number of previous failed tentative and the timestamp of the address. In this way, BitKer guarantees that each address will be contacted at least one time.

The Kernel uses two main threads: the *SocketListener* and the *KeepAlive* thread. The SocketListener manages the I/O communication to and from the Bitcoin network and has been implemented as a non-blocking I/O server. The KeepAlive thread is responsible to check the current connections, as explained in the following.

The SocketListener The *SocketListener* implements a non-blocking I/O server by using both a *ServerSocketChannel* to manage the connections, and a *Selector* that is responsible to manage the list of connections. When a connection is ready to be established, a new thread is run to manage the handshake phase, instead if the connection is not ready, the connection is aborted after a timeout fixed to 10 sec. Each input/output message is stored in buffers. The management of these buffers is critical because it has a strong impact on the use of the memory. For this reason, BitKer creates a buffer when it is needed by eliminating the overhead of a solution where a shared pool of buffers is concurrently accessed by a set of threads.

The KeepAlive thread This thread is activated every two minutes and it reads the list of known peers by checking their state. The thread can execute three actions:

- OPEN: when a connection with a peer is open, but there are not message available (sent or received) in the last minute, a ping message is sent to avoid the shut down of the connection;
- HANDSHAKE: when a connection has been in the handshake phase for at least 5 minutes, this means that the remote peer, does not accept the connection. The handshake phase is stopped and the client tries another handshake phase with an older version of the protocol;
- CLOSE: when a connection is closed, the timestamp is checked, and if it is older than 24 hours, it is removed from the list of known peers.

Choosing the IP addresses The strategy used to choose the IP addresses of the peers to be contacted in the initial phase is very important and represents the main choice for the definition of a discovery kernel client, like BitKer. Indeed, the amount of information gathered about the Bitcoin network depends on the number of peers which the kernel client is able to contact. The experiments conducted with BitKer have shown that the kernel client during its lifecycle is able to gather about 250000 addresses. The optimal case is when a client is able to connect to all the online peers, chosen among these 250000. The number A of active nodes in the network, is estimated as about 10000 nodes[3], so we can estimate that less than 10% of the 250000 known nodes are actually online. Moreover, it is important to consider that a connected node can drop the connection in every instant of time, because of its voluntary disconnection, of an error which as occurred or because BitKer has been banned by the remote connected peer. For these reasons, BitKer looks for new connections among the known 250000 addresses, every minute, chosen by a random strategy. To increase the number of connected nodes, the number of disconnections has to be less than the number of connections the client is able to establish every minute. In BitKer, the client tries to connect to 1250 different IP addresses, every minute. This value has been chosen so that the client does not run out of resources and the overhead of opening and closing socket connections does not impact on its performance. On the other way round, we have checked that this value guarantees that the number of connections increases up to a limit where the number of connections is approximatively the same of the disconnections.

If we suppose that the number of active connection is n, the average number of active nodes found among the 1250 chosen, is equal to:

$$\frac{1250 * (A - n)}{250000 - n} \tag{A.1}$$

[3] `https://bitnodes.21.co/`

This number decreases when the number of active connections increases. However, at the same time, the more connections are active, the more it is likely that one of these drops out.

A.2.2 The External Interface

BitKer is a core of a Bitcoin client and the External Interface provides live information to external programs who uses it. The External Interface is implemented by exploiting the publish-subscribe pattern [103]: messages are not directly sent when they are received, indeed they are collected in group of messages so that each subscriber receives the type of messages it is interested to. Each external program that communicates with BitKer generates one or more subscribers. The main components of the interface are two:

- *Event Service*. This component is a singleton and it is implemented as suggested in [127]. To avoid concurrency problems, the component creates a copy of the subscribers in a blocking-way when an event is detected. Then, the event is managed in a non-blocking way on the copy created before. This solution generates an overhead of copies but guarantees the concurrency.

- *Public Interface*. This component is a server for external programs that want to use BitKer as client. This component is responsible to manage connections in a non-blocking way, and to read requests sent by external programs. Moreover, it generates subscribers and sends reply messages.

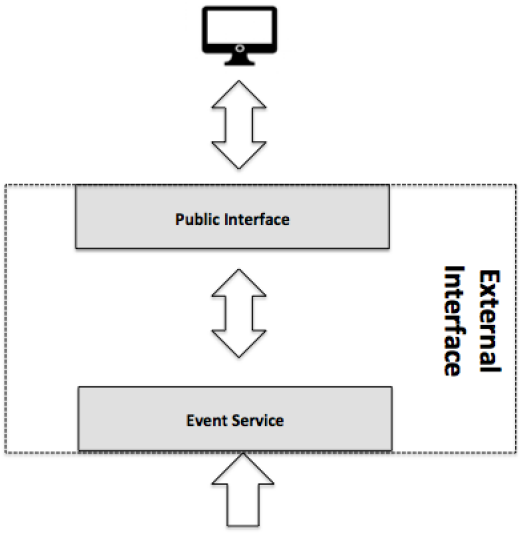


Figure A.5: The External Interface of BitKer

A.3 The BiKer API

BitKer defines an API for external applications which request operations. According to the publish-subscribe pattern, the client subscribes to one or more events by sending to BitKer one or more messages over a TCP connection. When an event occurs, BitKer publishes it by sending a reply message to the external clients which have subscribed.

Each message includes 4 bytes for the payload length, 4 bytes for the type of the request, and 8 bytes for the ID of the request. It is up to the client to uniquely identify each request, by inserting an unique ID in the request message. When BitKer receives a request, an *ACK* message is sent, followed by one or more reply messages. The ID received in the request is included in every reply message to recognize the corresponding request.

The messages used by BitKer are the following ones:

- **ListenTo**. The external client notifies its interest in a set of messages of the P2P protocol.
- **ListenFrom**. The external client notifies its interest in all the messages of the P2P protocol received by a particular set of IP addresses. (the IP address may not be connected at the time of the request in such case an attempt to establish a connection is made by BitKer)
- **SendTo**. The external client asks to send a message to a specific IP address (the IP address may not be connected at the time of the request in such case an attempt to establish a connection is made by BitKer).
- **SendToAll**. The external client asks to send a message to a number n of connected peers, chosen by BitKer
- **Terminate**. The message is used to stop a previous request identified by its ID
- **PeerStateChange**. The external client asks to be informed when BitKer detects a change in the status of any peer. The request message includes a code identifying the interested changes: 0 for all changes, 1 for the detection of a finished handshake phase, 2 for a disconnection and 3 when a phase of handshake begins.
- **Connection**. The external client asks to be informed when a new connection is opened.
- **List**. The external client request the list of all the peers connected to BitKer
- **ListenOut**. the external client asks to be informed about any message sent by BitKer.

Each reply message contains the message length, the type of message, and the ID of the request. The list of reply messages is:

- **Acknowledge**. This message is sent as reply to every request and includes the ID of the request, and a reply code specifying the correctness of the request.
- **MessageReceived**. This message is sent as reply to the *ListenTo* or *ListenFrom* requests and includes the message received by BitKer and the IP address of the sender.
- **MessageSent**. This message is an acknowledgment to the *SendTo* or *SendToAll* requests.
- **PeerList**. This message is sent as reply to the *List* message and it contains information about connected nodes at the time the request has been received. The body of the message contains an array of peers. Each peer maintains information such as: the IP address of the node, the service, who has started the connection and the user-agent of the peer.
- **State**. This message is sent as reply to *SentTo*, *ListenFrom*, and *PeerStateChange* requests. This message notifies the external client when a state of a peer changes.
- **Connect**. This message is sent as reply to *SendTo* and *ListenFrom* requests and notifies the outcome of the connection attempt that may be triggered from such requests.
- **NewConnection**. This message is sent in response to a *Connection* requests and notifies the external client when a new connection is established indicating also who has started the connection.
- **NewMessageSent**. This message is sent as reply to a *ListenOut* and it is used to indicate that a new message is sent.

A.4 Evaluation

BitKer is a kernel client which can be used for several purposes, the main one being the deanonymization of Bitcoin nodes. The main goal of our initial set of experiments is to test the effectiveness of BitKer in gathering data from the P2P network and in recognizing useful information. For this reason, we retrieved different datasets from the network to analyse them. The first dataset has been gathered in the period of time from 13th of April to 20th of April 2017. The second dataset has been gathered during the next week, from the 21th of April to 28th of April 2017. The third dataset has been gathered the week following the famous *Bitcoin Cash*

fork, from the 8th of August to 11th August 2017. We obtained more than 130GB of data.

In the following, we present the first set of analyses that enables the generation of a snapshot of the P2P network in terms of types of Bitcoin clients present in it. Moreover, we show several statistics related to the stability of the network, to the amount of traffic on the network, and finally to the way people use the Bitcoin clients.

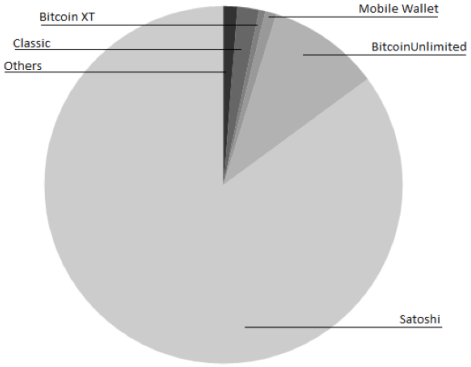
A.4.1 Analysis of client types

Figure A.6 shows the total number of clients, classified according to their type, in the three different datasets. As we can see, all the dataset show that more than 75% are official clients, this means that usually, the protocol used in the network is that of the Satoshi client, which fixes to 1MB the maximum limit of a block. It is interesting to note that, after the *Bitcoin Cash* fork, the number of unofficial client noticeably increases and becomes about 15% of the total. The clients that are the responsible of this increase are *BUCash* and *BitcoinABC* that are able to accept Bitcoin Cash transactions and blocks. It is worth noticing that our analysis has been done after one week from the launch of *Bitcoin cash*: in only one week, the number of nodes that use the new cryptocurrency, has grown till the 10% of the network. Another important observation, concerning the use of mobile wallets, is that their percentage is low despite the fact that is estimated to be most of the network clients. This maybe due to different factors. Mobile wallets don't accept connections in input so we must be chosen among the 8 clients they connect to, and the probability to be chosen is very low.

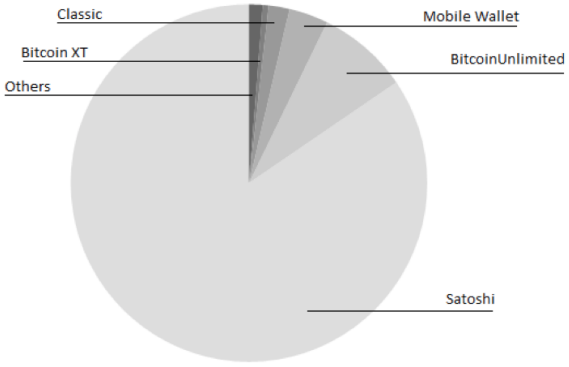
The category "Other" collects all unknown clients, like BitKer itself. In this category, we can find *bitcoin-ruby*, listeners such as *bitnodes*, but also customized client, such as *Nogtoshi*.

A.4.2 Analysis of the connection time

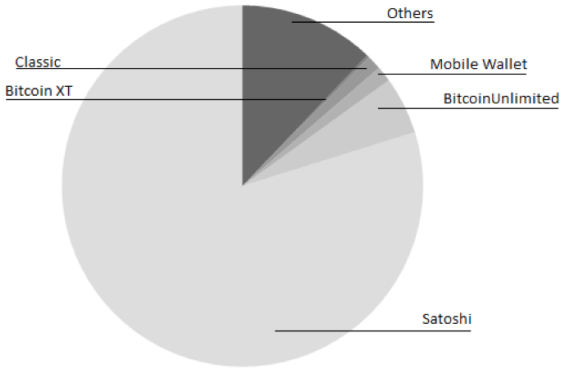
This set of analyses measures the average time each type of clients remained connected to BitKer. In Figure A.7, we can notice that the connection time of a classic client is lower than a client from the category Others, this is because Others includes nodes that intentionally remain always active, like BitKer, to gather data. We have observed that the behaviour of full nodes is similar: an active connection drops when an external event occur or when the client misbehaves: each client assigns to the connected peers a misbehavior score. Each time a misbehavior is detected, a misbehavior point is assigned to that peer, and when the number is over a fixed threshold the connection is dropped out.



(a) Number of clients (13/04/2017 - 20/04/2017)



(b) Number of clients (21/04/2017 - 28/04/2017).

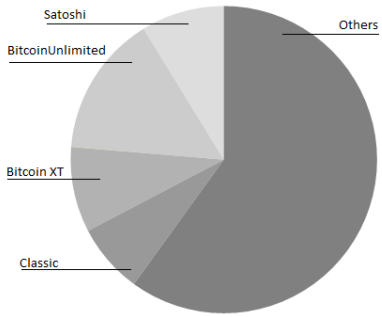


(c) Number of clients (04/08/2017 - 11/08/2017).

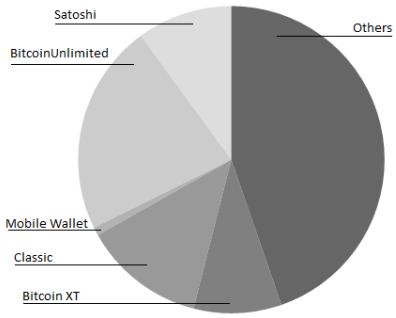
Figure A.6: Total Number of Bitcoin clients.

A.4.3 Analysis of the network traffic

The analysis of the network traffic evaluates the percentage of network traffic generated by each type of clients, measured as *INV messages* received by the different kind



(a) Average connection time (13/04/2017 - 20/04/2017)



(b) Average connection time (21/04/2017 - 28/04/2017).

Figure A.7: Average connection time for the Bitcoin clients.

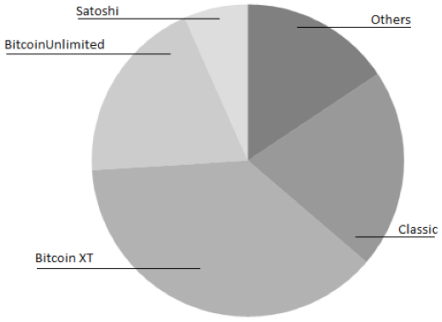
of clients. Figure A.8 shows that the Satoshi clients generate a very small amount of traffic, probably due to the fact that these clients connect to the network, send the transaction they are interested in and not relay others data. Instead most part of the traffic consists of messages sent by backbone clients, always active, and which are the core of the Bitcoin network.

In Figure A.9, we analyze in more detail the category Others. We can notice that there are noticeable differences in the number of messages each type of client sends. In particular, listeners do not send *INV messages* and are not reported in the Figure. Notice that listeners are connected to a huge amount of other nodes and cannot manage also the request and the transmission of *INV messages*.

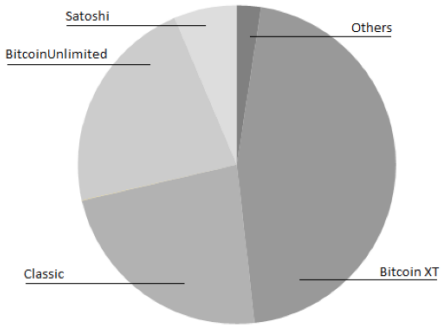
In Figure A.9(a), the bigger slice is related to the Cornell Falcon Network client. The goal of this client consists in broadcasting a new transaction in a faster way. However, in Figure A.9(b), we can observe an anomaly, because the client which has sent the major number of *INV* messages is the Statoshi client, which is a listener. This happens because the client is based on BitCore, which is a standard full node implementation of the Bitcoin protocol and it adds an analysis of the network[4].

Finally, the period shown in Figure A.9(c) does not provide a significant preva-

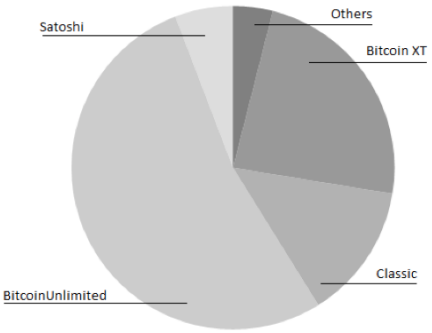
[4]`https://github.com/jlopp/statoshi`



(a) Average Number of Inventory message received (13/04/2017 - 20/04/2017)



(b) Average Number of Inventory message received (21/04/2017 - 28/04/2017).

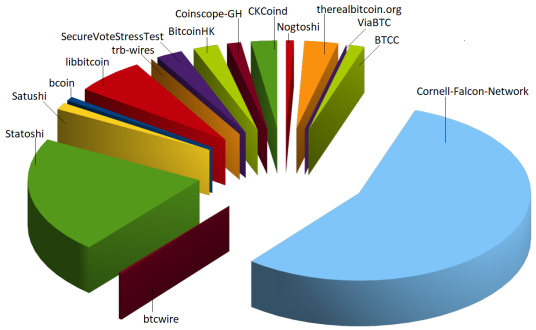


(c) Average Number of Inventory message received (04/08/2017 - 11/08/2017).

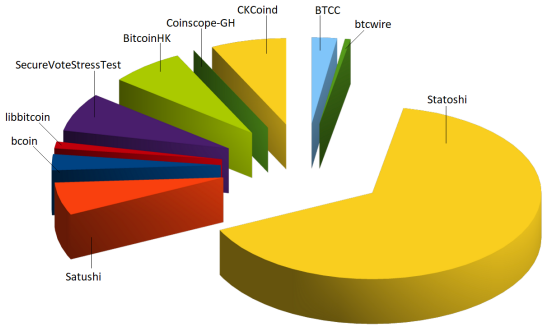
Figure A.8: Average Number of Inventory message received.

lence of a client to another. We can only consider that the CKCoind, Nogtoshi, Catoshi, and Satushi clients are the most predominant.

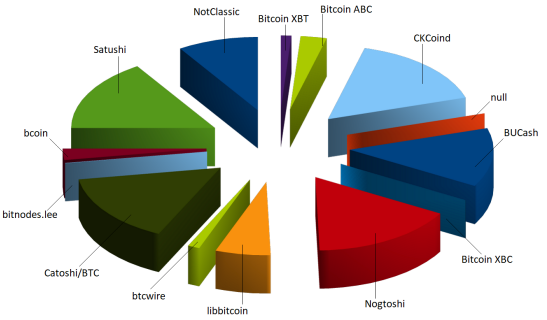
The last analysis we show concerns the comparison between the number of *INV* messages and the number of other messages (excluding the *INV* ones). In Table A.1, we propose, for each client, the total and average number of *INV* messages and other



(a) Average number of Inventory message for the category "Other" (13/04/2017 - 20/04/2017).



(b) Average number of Inventory messages for the category "Other" (21/04/2017 - 28/04/2017).



(c) Average number of Inventory messages for the category "Other" (04/08/2017 - 11/08/2017).

Figure A.9: Average Number of Inventory messages for the category "Others". Only the clients that sent at least one message are shown

messages. As we can see, the maximum amount of traffic in the network consists of *INV* messages. Instead, the other types of messages (ping, GETADDR, ADDR, etc..) represent a small part of the whole network traffic. This is a good signal that the network is active and stable because the traffic generated by the network is mainly due to the transactions and not to keep nodes connected. Moreover, by

observing the behaviour of listener clients, we can see that they don't send *INV* messages.

	Inventory		Other Messages	
	Total	Average	Total	Average
First Dataset (13/04/2018 - 20/04/2017)				
Satoshi	206164302	32818	6415284	1021
BitcoinUnlimited	71405642	95462	969489	1296
Mobile Wallet	0	0	97	4
Classic	14645237	101703	184998	1285
Bitcoin XT	7825531	186322	73021	1739
Others	7125067	67858	85024	810
Second Dataset (20/04/2018 - 28/04/2017)				
Satoshi	774307071	85521	22862154	2525
BitcoinUnlimited	262573262	297029	3739214	4230
Mobile Wallet	96364	324	12704	46
Classic	63081518	309223	827463	4056
Bitcoin XT	33638266	611605	320845	5834
Others	4336665	32607	620714	4667
Third Dataset (04/08/2018 - 08/08/2017)				
Satoshi	156662278	41503	20814568	2150
BitcoinUnlimited	92012550	379246	3783135	6082
Mobile Wallet	0	0	9491	55
Classic	5899827	97006	390711	2504
Bitcoin XT	1642447	168624	72717	2908
Others	16324960	28437	5047227	3428

Table A.1: Total Number of messages gathered during the three different periods.

A.5 Conclusions and Future work

In this chapter we presented BitKer, a Bitcoin kernel client able to connect to the Bitcoin network and retrieve information. BitKer provides an interface that can be used by external applications to interact with the Bitcoin network. The main characteristic of BitKer is that it is fast and provides a set of minimal functionalities that can be easily extended without changing the kernel client. Indeed, BitKer can interact with external applications with minimal effort.

We plan to extend BitKer in several directions. First of all, we plan to investigate further strategies for establishing connections with a larger set of peers. Furthermore, we plan to investigate more in detail the characteristics of the Bitcoin network by using BitKer. In particular, our final goal is to use the kernel client to implement a set of deanonymisation strategies whose goal is to pair a transaction with the node which generated it.

Bibliography

[1] AgoraVoting. `http://agoravoting.org/#index`. Accessed 31 Mar 2018. [Cited on page 122.]

[2] Rka Albert and Albert lszl Barabsi. Statistical mechanics of complex networks. *Rev. Mod. Phys*, 2002. [Cited on page 72.]

[3] Syed Taha Ali and Judy Murray. An overview of end-to-end verifiable voting systems. *Real-World Electronic Voting: Design, Analysis and Deployment*, pages 171–218, 2016. [Cited on page 118.]

[4] An Internet for Identity. `http://www.windley.com/archives/2016/08/an_internet_for_identity.shtml`. Accessed 31 Mar 2018. [Cited on pages vii and 127.]

[5] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. Evaluating user privacy in bitcoin. In *International Conference on Financial Cryptography and Data Security*, pages 34–51. Springer, 2013. [Cited on pages 44, 50, 52, 60, and 61.]

[6] Marcin Andrychowicz, Stefan Dziembowski, Daniel Malinowski, and Łukasz Mazurek. How to deal with malleability of bitcoin transactions. *arXiv preprint arXiv:1312.3230*, 2013. [Cited on page 48.]

[7] Suveen Angraal, Harlan M Krumholz, and Wade L Schulz. Blockchain technology: applications in health care. *Circulation: Cardiovascular Quality and Outcomes*, 10(9):e003800, 2017. [Cited on page 126.]

[8] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014. [Cited on pages v, 25, and 26.]

[9] Adam Back. A partial hash collision based postage scheme, 1997. [Cited on page 23.]

[10] Adam Back et al. Hashcash-a denial of service counter-measure. 2002. [Cited on pages 23 and 32.]

[11] Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. Four degrees of separation. In *Proceedings of the 4th Annual ACM Web Science Conference*, pages 33–42. ACM, 2012. [Cited on page 70.]

[12] Lear Bahack. Theoretical bitcoin attacks with less than half of the computational power (draft). *arXiv preprint arXiv:1312.7013*, 2013. [Cited on page 37.]

[13] Tobias Bamert, Christian Decker, Lennart Elsen, Roger Wattenhofer, and Samuel Welten. Have a snack, pay with bitcoins. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–5. IEEE, 2013. [Cited on page 42.]

[14] Banca d'Italia. Rapporto sulla stabilitá finanziaria, numero 1, maggio 2014. [Cited on page 23.]

[15] Silvia Bartolucci, Pauline Bernat, and Daniel Joseph. Sharvot: secret share-based voting on the blockchain. *arXiv preprint arXiv:1803.04861*, 2018. [Cited on page 122.]

[16] BBVA Research. Economic analysis bitcoin: A chapter in digital currency adoption, economic watch global, 2013. [Cited on page 23.]

[17] Be Your Own Bank. `https://techcrunch.com/2015/04/03/be-your-own-bank/`. Accessed 31 Mar 2018. [Cited on page 22.]

[18] Luca Becchetti, Paolo Boldi, Carlos Castillo, and Aristides Gionis. Efficient semi-streaming algorithms for local triangle counting in massive graphs. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–24. ACM, 2008. [Cited on page 72.]

[19] A. Berman and R. Plemmons. *Nonnegative Matrices in the Mathematical Sciences*. Society for Industrial and Applied Mathematics, 1994. [Cited on page 73.]

[20] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. Post-quantum cryptography.–2009. [Cited on page 26.]

[21] BIP, Bitcoin Improvement Proposals. `https://github.com/Bitcoin/bips`. Accessed 31 Mar 2018. [Cited on page 24.]

[22] Alex Biryukov, Dmitry Khovratovich, and Ivan Pustogarov. Deanonymisation of clients in bitcoin p2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 15–29. ACM, 2014. [Cited on pages 44, 45, 46, 47, and 111.]

[23] Alex Biryukov and Ivan Pustogarov. Bitcoin over tor isn't a good idea. In *Security and Privacy (SP), 2015 IEEE Symposium on*, pages 122–134. IEEE, 2015. [Cited on page 46.]

[24] Stefano Bistarelli, Marco Mantilacci, Paolo Santancini, and Francesco Santini. An end-to-end voting-system based on bitcoin. In *Proceedings of the Symposium on Applied Computing*, pages 1836–1841. ACM, 2017. [Cited on page 122.]

[25] Bitcoin CORE. `https://Bitcoin.org/en/version-history`. Accessed 31 Mar 2018. [Cited on page 24.]

[26] Bitcoin Developer Guide. `https://bitcoin.org/en/developer-guide#block-chain-overview`. Accessed 31 Mar 2018. [Cited on pages v and 30.]

[27] Bitcoin Wiki. `https://en.Bitcoin.it/wiki/Main_Page`. Accessed 31 Mar 2018. [Cited on pages 24, 95, and 97.]

[28] Bitcoin Wiki. retrieved 24 Feb 2017,`https://en.bitcoin.it/wiki/Transaction_fees`. [Cited on page 154.]

[29] Bitcoin Wiki. retrieved 24 Feb 2017,`https://en.bitcoin.it/wiki/Colored_Coins`. [Cited on page 155.]

[30] Bitcoin Wiki, Satoshi Dice. `https://en.bitcoin.it/wiki/Satoshi_Dice`. Accessed 31 Mar 2018. [Cited on pages vii, 96, and 110.]

[31] bitcointalk. `https://bitcointalk.org/index.php?topic=458934`. Accessed 31 Mar 2018. [Cited on page 95.]

[32] Bitcointalk.org, How Satoshidice is/can launder coins. `https://bitcointalk.org/index.php?topic=162807.msg1713104#msg1713104`. Accessed 31 Mar 2018. [Cited on page 111.]

[33] BitCongress Whitepaper. `https://bravenewcoin.com/assets/Whitepapers/BitCongressWhitepaper.pdf`. Accessed 31 Mar 2018. [Cited on page 121.]

[34] BitId. `https://github.com/bitid/bitid/blob/master/BIP_draft.md`. Accessed 31 Mar 2018. [Cited on page 130.]

[35] BITNODES. `https://getaddr.bitnodes.io/`. Accessed 31 Mar 2018. [Cited on pages 44 and 45.]

[36] BitShares. `https://bitshares.org/`. Accessed 31 Mar 2018. [Cited on page 121.]

[37] S Blake-Wilson and M Qu. Standards for efficient cryptography (sec) 2: Recommended elliptic curve domain parameters. *Certicom Research, Oct*, 1999. [Cited on page 25.]

[38] Blockchain.info Hashrate Distribution. `https://blockchain.info/pools?timespan=4days`. Accessed 24 Apr 2018. [Cited on pages v and 41.]

[39] Blockchains for supply chains part II. `http://resolvesp.com/blockchains-supply-chains-part-ii/`. Accessed 31 Mar 2018. [Cited on pages vii, 134, and 135.]

[40] Blockverify. `http://www.blockverify.io/`. Accessed 31 Mar 2018. [Cited on page 136.]

[41] Thomas Bocek, Bruno B Rodrigues, Tim Strasser, and Burkhard Stiller. Blockchains everywhere-a use-case of blockchains in the pharma supply-chain. In *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*, pages 772–777. IEEE, 2017. [Cited on page 137.]

[42] P. Boldi and S. Vigna. The webgraph framework i: Compression techniques. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 595–602. ACM, 2004. [Cited on page 69.]

[43] Paolo Boldi, Marco Rosa, and Sebastiano Vigna. Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In *Proceedings of the 20th international conference on World wide web*, pages 625–634. ACM, 2011. [Cited on page 70.]

[44] Paolo Boldi, Marco Rosa, and Sebastiano Vigna. Hyperanf: Approximating the neighbourhood function of very large graphs on a budget. In *Proceedings of the 20th international conference on World wide web*, pages 625–634. ACM, 2011. [Cited on page 87.]

[45] Paolo Boldi and Sebastiano Vigna. In-core computation of geometric centralities with hyperball: A hundred billion nodes and beyond. In *Proceedings of the 13th IEEE International Conference on Data Mining Workshops (ICDM)*, pages 621–628, 2013. [Cited on page 73.]

[46] Paolo Boldi and Sebastiano Vigna. Axioms for centrality. *Internet Mathematics*, 10(3-4):222–262, 2014. [Cited on pages 67 and 73.]

[47] Michele Borassi, David Coudert, Pierluigi Crescenzi, and Andrea Marino. On computing the hyperbolicity of real-world graphs. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, Patras, Greece, September 14-16, 2015, Proceedings*, pages 215–226, 2015. [Cited on page 106.]

[48] Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. On the solvability of the six degrees of kevin bacon game - A faster graph diameter and radius computation method. In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island,*

Sicily, Italy, July 1-3, 2014. Proceedings, pages 52–63, 2014. [Cited on page 70.]

[49] Michele Borassi, Pierluigi Crescenzi, Michel Habib, Walter A. Kosters, Andrea Marino, and Frank W. Takes. On the solvability of the six degrees of kevin bacon game - A faster graph diameter and radius computation method. In *Fun with Algorithms - 7th International Conference, FUN 2014, Lipari Island, Sicily, Italy, July 1-3, 2014. Proceedings*, pages 52–63, 2014. [Cited on pages 87 and 104.]

[50] C Brodersen, B Kalis, C Leong, E Mitchell, E Pupo, A Truscott, and LLP Accenture. Blockchain: Securing a new health interoperability experience, 2016. [Cited on page 124.]

[51] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*, 2017. [Cited on page 55.]

[52] Christian Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016. [Cited on page 135.]

[53] Kim Cameron. A user-centric identity metasystem. 2008. [Cited on page 127.]

[54] Blockchain Info Charts. `https://blockchain.info/charts/`. [Cited on page 60.]

[55] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983. [Cited on pages 23 and 120.]

[56] David L Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981. [Cited on pages 119 and 120.]

[57] D.P. Cheung and M. H. Gunes. A complex network analysis of the united states air transportation. In *Proceedings IEEE/ACM ASONAM, Washington, DC*, pages 699–701, 2012. [Cited on page 59.]

[58] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM Journal on Computing*, 14(1):210–223, 1985. [Cited on page 72.]

[59] Nicolas Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224. ACM, 2013. [Cited on page 23.]

[60] Chronicled. https://chronicled.com/. Accessed 31 Mar 2018. [Cited on page 136.]

[61] Flávio C Coelho. Optimizing disease surveillance by reporting on the blockchain. *bioRxiv*, page 278473, 2018. [Cited on page 126.]

[62] Coin Schedule, Cryptocurrency ICO Stats. https://www.coinschedule.com/stats.html. Accessed 31 Mar 2018. [Cited on page 3.]

[63] Coindesk, Walmart Blockchain Pilot Aims to Make China's Pork Market Safer. https://www.coindesk.com/walmart-blockchain-pilot-china-pork-market/. Accessed 31 Mar 2018. [Cited on page 135.]

[64] Counterparty. https://counterparty.io/. Accessed 31 Mar 2018. [Cited on page 121.]

[65] Nicolas T Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014. [Cited on page 40.]

[66] Crisis Strategy Draft, MtGox Unredacted. https://techcrunch.com/2014/02/25/here-is-the-unredacted-financials-page-from-the-mt-gox-presentation/. Accessed 31 Mar 2018, 2014. [Cited on page 49.]

[67] Jason Paul Cruz and Yuichi Kaji. E-voting system based on the bitcoin protocol and blind signatures. [Cited on page 122.]

[68] Current Standard for Dust Limit. retrieved 24 Feb 2017,https://github.com/bitcoin/bitcoin/blob/v0.10.0rc3/src/primitives/transaction.h#L137. [Cited on pages 74, 75, and 155.]

[69] W Dai. b-money. http://www.weidai.com/bmoney.txt. Accessed 31 Mar 2018, 1998. [Cited on page 23.]

[70] George Danezis, Cedric Fournet, Markulf Kohlweiss, and Bryan Parno. Pinocchio coin: building zerocoin from a succinct pairing-based proof system. In *Proceedings of the First ACM workshop on Language support for privacy-enhancing technologies*, pages 27–30. ACM, 2013. [Cited on page 44.]

[71] Dapp-bin: A place for all the dApps to live, Collection of dApps Examples. https://github.com/ethereum/dapp-bin. Accessed 31 Mar 2018. [Cited on page 56.]

[72] Christian Decker. CABRA, Bitcoin research at your fingertips! https://cdecker.github.io/btcresearch/. Accessed 31 Mar 2018. [Cited on pages v and 2.]

[73] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*, pages 1–10. IEEE, 2013. [Cited on pages 33, 42, and 44.]

[74] Christian Decker and Roger Wattenhofer. Bitcoin transaction malleability and mtgox. In *European Symposium on Research in Computer Security*, pages 313–326. Springer, 2014. [Cited on pages 47, 48, 49, and 111.]

[75] Democracy.earth. `http://democracy.earth/`. Accessed 31 Mar 2018. [Cited on page 122.]

[76] Giuseppe Di Battista, Valentino Di Donato, Maurizio Patrignani, Maurizio Pizzonia, Vincenzo Roselli, and Roberto Tamassia. Bitconeview: visualization of flows in the bitcoin transaction graph. In *Visualization for Cyber Security (VizSec), 2015 IEEE Symposium on*, pages 1–8. IEEE, 2015. [Cited on page 84.]

[77] D Di Francesco Maesa, A. Marino, and L. Ricci. Uncovering the bitcoin blockchain: an analysis of the full users graph. In *IEEE DSAA 2016, 3rd IEEE International Conference on Data Science and Advanced Analytics, Montreal, October*, 2016. [Cited on page 77.]

[78] Damiano Di Francesco Maesa. Bitcoin protocol main threats. Technical Report, Computer Science Department. University of Pisa, Pisa, IT. `http://eprints.adm.unipi.it/2371/1/TechRep.pdf`, 2017. [Cited on page xiv.]

[79] Damiano Di Francesco Maesa, Matteo Franceschi, Barbara Guidi, and Laura Ricci. Bitker: a p2p kernel client for bitcoin. In *7th International Workshop on Peer-to-Peer Architectures, Networks and Systems (PANS 2018). As part of The 16th International Conference on High Performance Computing & Simulation (HPCS 2018). To Appear*, 2018. [Cited on page xiii.]

[80] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. An analysis of the bitcoin users graph: inferring unusual behaviours. In *International Workshop on Complex Networks and their Applications*, pages 749–760. Springer International Publishing, 2016. [Cited on page xiii.]

[81] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Uncovering the bitcoin blockchain: an analysis of the full users graph. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*, pages 537–546. IEEE, 2016. [Cited on page xiii.]

[82] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Data-driven analysis of bitcoin properties: exploiting the users graph. *International Journal of Data Science and Analytics*, pages 1–18, 2017. [Cited on pages xiii and 7.]

[83] Damiano Di Francesco Maesa, Andrea Marino, and Laura Ricci. Detecting artificial behaviours in the bitcoin users graph. *Online Social Networks and Media*, 3:63–74, 2017. [Cited on page xiii.]

[84] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control. In *IFIP International Conference on Distributed Applications and Interoperable Systems*, pages 206–220. Springer, 2017. [Cited on pages xiii, 55, and 167.]

[85] Damiano Di Francesco Maesa, Paolo Mori, and Laura Ricci. Blockchain based access control services. In *IEEE Symposium on Recent Advances on Blockchain and its Applications, 2018 IEEE International Conference on Blockchain*, 2018. [Cited on page xiii.]

[86] Damiano Di Francesco Maesa, Laura Ricci, and Paolo Mori. distributed access control through blockchain technology. *Blockchain Engineering*, page 31, 2017. [Cited on page xiii.]

[87] Diamond Blockchain Initiative. `https://www.diamondblockchaininitiative.com/`. Accessed 31 Mar 2018. [Cited on page 135.]

[88] Caribou Digital. Private-sector digital identity in emerging markets farnham, surrey, united kingdom: Caribou digital publishing. 2016. [Cited on pages 127 and 130.]

[89] Hans Dobbertin, Antoon Bosselaers, and Bart Preneel. Ripemd-160: A strengthened version of ripemd. In *International Workshop on Fast Software Encryption*, pages 71–82. Springer, 1996. [Cited on page 26.]

[90] Sander Dommers, Remco Van Der Hofstad, and Gerard Hooghiemstra. Diameters in preferential attachment models. *Journal of Statistical Physics*, 139(1):72–107, 2010. [Cited on pages 83 and 87.]

[91] Joan Antoni Donet Donet, Cristina Pérez-Sola, and Jordi Herrera-Joancomartí. The bitcoin p2p network. In *International Conference on Financial Cryptography and Data Security*, pages 87–102. Springer, 2014. [Cited on pages 44 and 45.]

[92] Dravis Group. Bitcoin, digital currency and the internet of money. Dravis Group LLC, San Francisco, USA, 2014. [Cited on page 23.]

[93] Alevtina Dubovitskaya, Zhigang Xu, Samuel Ryu, Michael Schumacher, and Fusheng Wang. Secure and trustable electronic medical records sharing using blockchain. *arXiv preprint arXiv:1709.06528*, 2017. [Cited on page 126.]

[94] Paul Dunphy and Fabien AP Petitcolas. A first look at identity management schemes on the blockchain. *arXiv preprint arXiv:1801.03294*, 2018. [Cited on page 129.]

[95] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference*, pages 139–147. Springer, 1992. [Cited on page 32.]

[96] Ariel Ekblaw, Asaph Azaria, John D Halamka, and Andrew Lippman. A case study for blockchain in healthcare:medrec prototype for electronic health records and medical research data. In *Proceedings of IEEE Open & Big Data Conference*, volume 13, page 13, 2016. [Cited on page 126.]

[97] Craig Kent Elwell, M Maureen Murphy, Michael V Seitzinger, and Edward Vincent Murphy. Bitcoin: questions, answers, and analysis of legal issues, 2013. [Cited on page 23.]

[98] Adam Kaleb Ernest. The key to unlocking the black box: Why the world needs a transparent voting dac. [Cited on page 121.]

[99] H. Es-Samaali, A. Outchakoucht, and J.P. Leroy. A blockchain-based access control for big data. *International Journal of Computer Networks and Communications Security*, 5(7):137–147, 2017. [Cited on page 141.]

[100] Ethash - Ethereum Wiki. `https://github.com/ethereum/wiki/wiki/Ethash`. [Cited on page 55.]

[101] Ethereum Improvement Proposal (EIP) 1011: Hybrid Casper FFG. `https://eips.ethereum.org/EIPS/eip-1011`. Accessed 31 Mar 2018. [Cited on page 55.]

[102] Etherscan Ethereum Main Chain. `https://etherscan.io/`. Accessed 31 Mar 2018. [Cited on pages viii and 195.]

[103] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003. [Cited on page 214.]

[104] European Central Bank. Virtual currency schemes, october 2012. [Cited on page 23.]

[105] Everledger. `https://www.everledger.io/`. Accessed 31 Mar 2018. [Cited on page 135.]

[106] Everledger, Diamonds. `https://diamonds.everledger.io/`. Accessed 31 Mar 2018. [Cited on page 136.]

[107] Evolution of blockchain technology Insights from the GitHub platform. `https://www2.deloitte.com/insights/us/en/industry/financial-services/evolution-of-blockchain-github-platform.html`. Accessed 31 Mar 2018. [Cited on pages v, 2, and 3.]

[108] Wallet Explorer. `https://www.walletexplorer.com/`. [Cited on pages 66 and 110.]

[109] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer, 2014. [Cited on pages v, 37, 38, and 39.]

[110] Factom. `https://www.factom.com/`. Accessed 31 Mar 2018. [Cited on page 136.]

[111] FBI. Bitcoin virtual currency: Unique features present distinct challenges for deterring illicit activity. ech. rep., Federal Bureau of Investigation, 2012. [Cited on page 23.]

[112] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of cryptology*, 1(2):77–94, 1988. [Cited on pages 122 and 136.]

[113] Reid Fergal and Martin Harrigan. An analysis of anonymity in the bitcoin system. In *Proceeding of 2011 PASSAT/SocialCom 2011*, pages 1318–1326. IEEE, 2011. [Cited on pages v, 7, 44, 45, 49, 50, 52, and 61.]

[114] Finney Attack. `https://Bitcointalk.org/index.php?topic=3441.msg48384#msg48384`. Accessed 31 Mar 2018. [Cited on page 42.]

[115] Michael J Fischer, Nancy A Lynch, and Michael S Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM (JACM)*, 32(2):374–382, 1985. [Cited on page 33.]

[116] FollowMyVote. `https://followmyvote.com/`. Accessed 31 Mar 2018. [Cited on page 121.]

[117] Four Years of Token Sales, Visualized in One Graphic. `https://elementus.io/blog/token-sales-visualization/`. Accessed 31 Mar 2018. [Cited on page 3.]

[118] Zhimin Gao, Lei Xu, Lin Chen, Xi Zhao, Yang Lu, and Weidong Shi. Coc: A unified distributed ledger based supply chain management system. *Journal of Computer Science and Technology*, 33(2):237–248, 2018. [Cited on page 137.]

[119] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015. [Cited on page 33.]

[120] David Garcia, Claudio J Tessone, Pavlin Mavrodiev, and Nicolas Perony. The digital traces of bubbles: feedback cycles between socio-economic signals in the bitcoin economy. *Journal of the Royal Society Interface*, 11(99):20140623, 2014. [Cited on page 84.]

[121] Gem. `https://gem.co/health/`. Accessed 31 Mar 2018. [Cited on page 126.]

[122] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford University, 2009. [Cited on pages 125 and 136.]

[123] Arthur Gervais, Srdjan Capkun, Ghassan O Karame, and Damian Gruber. On the privacy provisions of bloom filters in lightweight bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 326–335. ACM, 2014. [Cited on page 54.]

[124] Arthur Gervais, Hubert Ritzdorf, Ghassan O. Karame, and Srdjan Capkun. Tampering with the delivery of blocks and transactions in bitcoin. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 692–705, New York, NY, USA, 2015. ACM. [Cited on page 210.]

[125] geth client. `https://github.com/ethereum/go-ethereum/wiki/geth`. [Cited on page 172.]

[126] Google Trends. `https://trends.google.com/trends/`. Accessed 31 Mar 2018. [Cited on pages vii and 118.]

[127] Suchitra Gupta, Jeffrey M Hartkopf, and Suresh Ramaswamy. Event notifier: a pattern for event notification. In *More Java gems*, pages 131–153. Cambridge University Press, 2000. [Cited on page 214.]

[128] M.H. Gunes H. Kardes, A. Sevincer and M. Yuksel. In *Six degrees of separation among US researchers Proceedings of IEEE/ACM SONAM*, pages 654–659, 2012. [Cited on page 59.]

[129] M. Harrigan and C. Fretter. The unreasonable effectiveness of address clustering. In *13th IEEE International Conference on Advanced and Trusted Computing (ATC'16)*, 2016. [Cited on page 61.]

[130] Hashed Health. `http://www.hashedhealth.com/`. Accessed 31 Mar 2018. [Cited on page 126.]

[131] Healthcare Working Group. https://www.hyperledger.org/industries/healthcare. Accessed 31 Mar 2018. [Cited on page 126.]

[132] Hedera Hasgraph. https://www.hederahashgraph.com/. Accessed 31 Mar 2018. [Cited on page 6.]

[133] Thomas Heston. A case study in blockchain healthcare innovation. 2017. [Cited on page 126.]

[134] Hidden surprises in the Bitcoin blockchain. retrieved 24 Feb 2017,http://www.righto.com/2014/02/ascii-bernanke-wikileaks-photographs.html. [Cited on page 153.]

[135] Lars Holdgaard. An exploration of the bitcoin ecosystem. *Bitcoin-Expert. net*, 2014. [Cited on page 23.]

[136] Nicolas Houy. It will cost you nothing to'kill'a proof-of-stake crypto-currency. 2014. [Cited on page 34.]

[137] Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C Snoeren, and Kirill Levchenko. Botcoin: Monetizing stolen cycles. In *NDSS*. Citeseer, 2014. [Cited on page 34.]

[138] Hyperledger Burrow Project. https://github.com/hyperledger/burrow. Accessed 31 Mar 2018. [Cited on page 57.]

[139] Mihaela Iavorschi et al. The bitcoin project and the free market. *CES Working Papers*, 5(4):529–534, 2013. [Cited on page 23.]

[140] IdchainZ. http://idchainz.com/. Accessed 31 Mar 2018. [Cited on page 130.]

[141] ID.me. https://www.id.me/. Accessed 31 Mar 2018. [Cited on page 130.]

[142] Imogen Heap, Thiny Human. https://imogen2.surge.sh/#/imogen_heap/tiny_human/tiny_human. Accessed 31 Mar 2018. [Cited on page 132.]

[143] Inno.Vote Whitepaper. http://inno.vote/whitepaper/Inno.vote%20%E2%80%94%20Bringing%20Democracy%20to%20Elections.pdf. Accessed 31 Mar 2018. [Cited on page 122.]

[144] Internal Revenue Service (IRS), IRS Notice 2014-21, 2014. [Cited on page 23.]

[145] IOTA. https://iota.org/. Accessed 31 Mar 2018. [Cited on page 6.]

[146] IPFS. https://ipfs.io/. Accessed 31 Mar 2018. [Cited on page 129.]

[147] Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM Journal on Computing*, 7(4):413–423, 1978. [Cited on page 72.]

[148] Drew Ivan. Moving toward a blockchain-based method for the secure storage of patient records. In *ONC/NIST Use of Blockchain for Healthcare and Research Workshop. Gaithersburg, Maryland, United States: ONC/NIST*, 2016. [Cited on page 123.]

[149] Ori Jacobovitz. Blockchain for identity management, 2016. [Cited on page 130.]

[150] White MA K. Komurov, MH.Gunes. Fine-scale dissection of functional protein network organization by statistical network analysis. *PLoS ONE*, 4(6), 2009. [Cited on page 59.]

[151] Ghassan Karame, Elli Androulaki, and Srdjan Capkun. Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. *IACR Cryptology ePrint Archive*, 2012(248), 2012. [Cited on page 42.]

[152] Anne-Marie Kermarrec and Maarten Van Steen. Gossiping in distributed systems. *ACM SIGOPS Operating Systems Review*, 41(5):2–7, 2007. [Cited on page 208.]

[153] Sunny King. Primecoin: Cryptocurrency with prime number proof-of-work. *July 7th*, 2013. [Cited on page 34.]

[154] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012. [Cited on page 34.]

[155] kevin Kirby, Anthony Masi, and Fernando Maymi. Votebook a proposal for a blockchain-based electronic voting system. [Cited on page 122.]

[156] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one*, 9(2):e86197, 2014. [Cited on pages 60, 61, 63, and 71.]

[157] Kari Korpela, Jukka Hallikas, and Tomi Dahlberg. Digital supply chain transformation toward blockchain integration. In *proceedings of the 50th Hawaii international conference on system sciences*, 2017. [Cited on page 137.]

[158] Philip Koshy, Diana Koshy, and Patrick McDaniel. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security*, pages 469–485. Springer, 2014. [Cited on pages 44, 45, and 111.]

[159] Kovan Ethereum Testnet. `https://github.com/kovan-testnet/proposal`. Accessed 31 Mar 2018. [Cited on page 172.]

[160] Joshua A Kroll, Ian C Davey, and Edward W Felten. The economics of bitcoin mining, or bitcoin in the presence of adversaries. In *Proceedings of WEIS*, volume 2013, page 11, 2013. [Cited on page 36.]

[161] Nir Kshetri. 1 blockchain's roles in meeting key supply chain management objectives. *International Journal of Information Management*, 39:80–89, 2018. [Cited on page 137.]

[162] Tsung-Ting Kuo and Lucila Ohno-Machado. Modelchain: Decentralized privacy-preserving healthcare predictive modeling framework on private blockchain networks. *arXiv preprint arXiv:1802.01746*, 2018. [Cited on page 125.]

[163] Ulrich Lang. Openpmf scaas: Authorization as a service for cloud & SOA applications. In *Second International Conference on Cloud Computing (CloudCom 2010)*, pages 634–643, 2010. [Cited on page 140.]

[164] A. Lazouski, F. Martinelli, and P. Mori. A prototype for enforcing usage control policies based on XACML. In *Trust, Privacy and Security in Digital Business. TrustBus 2012. Lecture Notes in Computer Science, vol 7449*, pages 79–92. Springer-Verlag Berlin Heidelberg, 2012. [Cited on page 139.]

[165] Kibin Lee, Joshua I James, Tekachew Gobena Ejeta, and Hyoung Joong Kim. Electronic voting service using block-chain. *The Journal of Digital Forensics, Security and Law: JDFSL*, 11(2):123, 2016. [Cited on page 122.]

[166] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187. ACM, 2005. [Cited on pages 70 and 87.]

[167] Laure A Linn and Martha B Koo. Blockchain for health data and its potential use in health it and health care related research. In *ONC/NIST Use of Blockchain for Healthcare and Research Workshop, Gaithersburg, Maryland, United States: ONC/NIST*, 2016. [Cited on page 124.]

[168] Mattias Lischke and Benjamin Fabian. Analyzing the bitcoin network: The first four years. *Future Internet*, 8(1), 2016. [Cited on pages 60 and 61.]

[169] Litecoin. `https://litecoin.info/Main_Page`. Accessed 31 Mar 2018. [Cited on page 34.]

[170] LLL PoC 6. `https://github.com/ethereum/cpp-ethereum/wiki/LLL-PoC-6/04fae9e627ac84d771faddcf60098ad09230ab58`. Accessed 31 Mar 2018. [Cited on page 57.]

[171] William J Luther and Josiah Olson. Bitcoin is memory. 2013. [Cited on page 23.]

[172] Maersk, Maersk and IBM to form joint venture applying blockchain to improve global trade and digitise supply chains. `https://www.maersk.com/press/press-release-archive/maersk-and-ibm-to-form-joint-venture`. Accessed 31 Mar 2018. [Cited on page 135.]

[173] D. Di Francesco Maesa, A. Marino, and L. Ricci. An analysis of the bitcoin users graph: inferring unusual behaviours. In *Proceedings of the 5-th International Workshop on Complex Networks and their Applications, Milan*, 2016. [Cited on page 70.]

[174] Omri Y Marian. Are cryptocurrencies 'super' tax havens? 2013. [Cited on page 23.]

[175] Daniel Martens and Walid Maalej. Reviewchain: Untampered product reviews on the blockchain. *arXiv preprint arXiv:1803.01661*, 2018. [Cited on page 137.]

[176] Martina Matta, Ilaria Lunesu, and Michele Marchesi. The predictor impact of web search media on bitcoin trading volumes. In *Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K), 2015 7th International Joint Conference on*, volume 1, pages 620–626. IEEE, 2015. [Cited on page 84.]

[177] CoinJoin Maxwell. Bitcoin privacy for the real worldcoinjoin: Bitcoin privacy for the real world, bitcointalk. org, 2013. [Cited on pages 45, 51, and 52.]

[178] Patrick McCorry, Siamak F Shahandashti, and Feng Hao. A smart contract for boardroom voting with maximum voter privacy. In *International Conference on Financial Cryptography and Data Security*, pages 357–375. Springer, 2017. [Cited on pages 121 and 122.]

[179] Dan McGinn, David Birch, David Akroyd, Miguel Molina-Solana, Yike Guo, and William J Knottenbelt. Visualizing dynamic bitcoin transaction patterns. *Big data*, 4(2):109–119, 2016. [Cited on pages 84 and 95.]

[180] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kitrill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013*, pages 127–140, 2013. [Cited on pages v, 45, 49, 50, 52, and 53.]

[181] Sarah Meiklejohn, Marjori Pomarole, Grant Jordan, Kitrill Levchenko, Damon McCoy, Geoffrey M. Voelker, and Stefan Savage. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the*

2013 Internet Measurement Conference, IMC 2013, Barcelona, Spain, October 23-25, 2013, pages 127–140, 2013. [Cited on pages 60, 61, 84, and 88.]

[182] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 369–378. Springer, 1987. [Cited on page 31.]

[183] MetaMask. `https://metamask.io/`. Accessed 31 Mar 2018. [Cited on page 191.]

[184] Matthias Mettler. Blockchain technology in healthcare: The revolution starts here. In *e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on*, pages 1–3. IEEE, 2016. [Cited on page 126.]

[185] Ian Miers, Christina Garman, Matthew Green, and Aviel D Rubin. Zerocoin: Anonymous distributed e-cash from bitcoin. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 397–411. IEEE, 2013. [Cited on page 44.]

[186] Andrew Miller and Joseph J LaViola Jr. Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin. *Available on line: http://nakamotoinstitute. org/research/anonymous-byzantine-consensus*, 2014. [Cited on page 33.]

[187] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin's public topology and influential nodes. *et al.*, 2015. [Cited on pages 45, 46, 47, and 111.]

[188] Modum. `https://modum.io/`. Accessed 31 Mar 2018. [Cited on page 137.]

[189] Monax Industries Limited. `https://monax.io/`. Accessed 31 Mar 2018. [Cited on page 57.]

[190] Monegraph. `https://monegraph.com/`. Accessed 31 Mar 2018. [Cited on page 132.]

[191] Malte Moser, Rainer Bohme, and Dominic Breuker. An inquiry into money laundering tools in the bitcoin ecosystem. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–14. IEEE, 2013. [Cited on pages 84 and 94.]

[192] Malte Moser. Anonymity of bitcoin transactions. 2013. [Cited on page 53.]

[193] MUSE. `https://museblockchain.com/`. Accessed 31 Mar 2018. [Cited on page 132.]

[194] Mutan GitHub Repository. `https://github.com/obscuren/mutan`. Accessed 31 Mar 2018. [Cited on page 57.]

[195] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. [Cited on pages v, 2, 4, 24, 29, 31, 35, 37, 43, 54, 59, and 61.]

[196] Yomna Nasser, Chidinma Okoye, Jeremy Clark, and Peter YA Ryan. Blockchains and voting: Somewhere between hype and a panacea. [Cited on page 121.]

[197] Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003. [Cited on page 105.]

[198] NIST, Description of SHA-256, SHA-384 and SHA-512. `http://csrc.nist.gov/groups/STM/cavp/documents/shs/sha256-384-512.pdf`. Accessed 31 Mar 2018. [Cited on page 26.]

[199] John Normand. The audacity of bitcoin - risks and opportunities for corporates and investors, 2014. [Cited on page 23.]

[200] OASIS. eXtensible Access Control Markup Language (XACML) version 3.0, January 2013. [Cited on pages 140, 141, 142, 143, 152, and 173.]

[201] Micha Ober, Stefan Katzenbeisser, and Kay Hamacher. Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250, 2013. [Cited on pages 50, 60, and 61.]

[202] Department of the Treasury Financial Crimes Enforcement Network. Guidance FIN-2013-G001 Issued: March 18, 2013 Subject: Application of FinCEN's Regulations to Persons Administering, Exchanging, or Using Virtual Currencies, 2013. [Cited on page 23.]

[203] Open Blockchain initiative. `http://blockchain.open.ac.uk/`. [Cited on page 172.]

[204] Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, 9(18):5943–5964, 2016. [Cited on page 141.]

[205] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999. [Cited on page 73.]

[206] Peercoin. `http://peercoin.net/`. Accessed 31 Mar 2018. [Cited on page 34.]

[207] Colin Percival. scrypt: A new key derivation function doing our best to thwart tlas armed with asics, 2009. [Cited on page 34.]

[208] Colin Percival. Stronger key derivation via sequential memory-hard functions. *Self-published*, pages 1–16, 2009. [Cited on page 34.]

[209] Colin Percival and Simon Josefsson. The scrypt password-based key derivation function. Technical report, 2016. [Cited on page 34.]

[210] Kevin Peterson, Rammohan Deeduvanu, Pradip Kanjamala, and Kelly Boles. A blockchain-based approach to health information exchange networks. In *Proc. NIST Workshop Blockchain Healthcare*, volume 1, pages 1–10, 2016. [Cited on page 124.]

[211] PokitDot. `http://www.pokitdok.com/`. Accessed 31 Mar 2018. [Cited on page 126.]

[212] Polys. `https://polys.me/`. Accessed 31 Mar 2018. [Cited on page 122.]

[213] M. K. Popuri and M. H. Gunes. empirical analysis of crypto currencies. In *7th Workshop on Complex Networks (CompleNet), Dijon, France, Mar 23-25*, 2016. [Cited on page 62.]

[214] Bart Preneel, Antoon Bosselaers, and Hans Dobbertin. The cryptographic hash function ripemd-160, 1997. [Cited on page 26.]

[215] Primecoin. `http://primecoin.org//`. Accessed 31 Mar 2018. [Cited on page 34.]

[216] Primecoin wiki. `https://github.com/primecoin/primecoin/wiki/_pages`. Accessed 31 Mar 2018. [Cited on page 34.]

[217] Procivis and University of Zurich develop blockchain based e-voting solution Procivis. `http://procivis.ch/2017/09/26/procivis-and-university-of-zurich-develop-blockchain-based-e-voting-solution/`. Accessed 31 Mar 2018. [Cited on page 122.]

[218] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *arXiv preprint quant-ph/0301141*, 2003. [Cited on page 26.]

[219] ProtocolBuffers. `https://developers.google.com/protocol-buffers/`. [Cited on page 66.]

[220] Provenance. `https://www.provenance.org`. Accessed 31 Mar 2018. [Cited on pages vii and 136.]

[221] Provenance, tracking tuna on the blockchain. `https://www.provenance.org/tracking-tuna-on-the-blockchain`. Accessed 31 Mar 2018. [Cited on page 136.]

[222] G. J. Ahn H. Sharifi R. Wu, X. Zhang and H. Xie. ACaaS: Access control as a service for iaas cloud. In *International Conference on Social Computing*, pages 423–428, 2013. [Cited on page 140.]

[223] R3 Corda. `https://docs.corda.net/`. Accessed 31 Mar 2018. [Cited on page 6.]

[224] Radix. `https://www.radixdlt.com/`. Accessed 31 Mar 2018. [Cited on page 6.]

[225] Re: Bitcoin P2P e-cash paper 2008-11-09 14:13:34 UTC. `http://satoshi.nakamotoinstitute.org/emails/cryptography/6/#selection-7.0-11.23`. Accessed 31 Mar 2018. [Cited on page 2.]

[226] reddit. `https://www.reddit.com/r/Bitcoin/comments/1xenyd/just_received_weird_tiny_payments_1sochi_1enjoy/`. Accessed 31 Mar 2018. [Cited on page 95.]

[227] Rinkeby Ethereum Testnet. `https://github.com/ethereum/EIPs/issues/225`. Accessed 31 Mar 2018. [Cited on pages 172 and 190.]

[228] Robomed Network Whitepaper. `https://robomed.io/download/Robomed_whitepaper_eng_final.pdf`. Accessed 31 Mar 2018. [Cited on page 126.]

[229] Denis Jaromil Roio. Bitcoin, the end of the taboo on money. *Dyne. org digital press (April 2013) https://files. dyne. org/readers/Bitcoin_end_of_taboo_on_money. pdf*, 2013. [Cited on page 23.]

[230] Dorit Ron and Adi Shamir. Quantitative analysis of the full bitcoin transaction graph. In *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, pages 6–24, 2013. [Cited on pages 45, 50, 52, 60, 61, 70, 84, and 94.]

[231] Dorit Ron and Adi Shamir. How did dread pirate roberts acquire and protect his bitcoin wealth? In *International Conference on Financial Cryptography and Data Security*, pages 3–15. Springer, 2014. [Cited on page 84.]

[232] Ropsetn Revival. `https://github.com/ethereum/ropsten`. Accessed 31 Mar 2018. [Cited on page 172.]

[233] Ropsten Ethereum Testnet. `https://ropsten.etherscan.io/`. Accessed 31 Mar 2018. [Cited on page 172.]

[234] Meni Rosenfeld. Analysis of hashrate-based double spending. *arXiv preprint arXiv:1402.2009*, 2014. [Cited on page 42.]

[235] Tim Ruffing, Pedro Moreno-Sanchez, and Aniket Kate. Coinshuffle: Practical decentralized coin mixing for bitcoin. In *European Symposium on Research in Computer Security*, pages 345–364. Springer, 2014. [Cited on pages 51, 52, 61, and 122.]

[236] Sablik, Tim. Digital Currency : New Private Currencies Like Bitcoin Offer Potential and Puzzles. Econ Focus, Third Quarter, pp. 18-27, 2013. `https://fraser.stlouisfed.org/scribd/?toc_id=503679&filepath=/files/docs/historical/frbrich/focus/frbrich_focus_201303.pdf&start_page=29`. Accessed 31 Mar 2018. [Cited on page 23.]

[237] Ravi S Sandhu and Pierangela Samarati. Access control: principle and practice. *Communications Magazine, IEEE*, 32(9):40–48, 1994. [Cited on page 139.]

[238] Eli Ben Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *Security and Privacy (SP), 2014 IEEE Symposium on*, pages 459–474. IEEE, 2014. [Cited on page 44.]

[239] Secure.vote. `https://secure.vote/`. Accessed 31 Mar 2018. [Cited on page 122.]

[240] Serpent GitHub Repository. `https://github.com/ethereum/serpent/tree/develop`. Accessed 31 Mar 2018. [Cited on page 57.]

[241] ShoCard. `https://shocard.com/how-it-works/`. Accessed 31 Mar 2018. [Cited on page 130.]

[242] ShoCard Whitepaper. `https://shocard.com/wp-content/uploads/2016/11/travel-identity-of-the-future.pdf`. Accessed 31 Mar 2018. [Cited on page 129.]

[243] SimplyVital Health. `https://www.simplyvitalhealth.com/`. Accessed 31 Mar 2018. [Cited on page 126.]

[244] Solidity 0.4.23 documentation, Types. `https://solidity.readthedocs.io/en/v0.4.23/types.html`. Accessed 31 Mar 2018. [Cited on page 57.]

[245] Solidity, Bitcoin Wiki. `https://en.m.bitcoinwiki.org/wiki/Solidity`. Accessed 31 Mar 2018. [Cited on page 57.]

[246] Solidity compiler releases. `https://github.com/ethereum/solidity/releases`. [Cited on page 177.]

[247] Solidity documentation. `https://solidity.readthedocs.io/en/develop/`. [Cited on page 57.]

[248] Solidity Preliminary Proposal. `https://stackedit.io/viewer#!url=https://gist.githubusercontent.com/gavofyork/31b35cd2252a00d0d057/raw/16de06189d2175d2e31b300f1f8531e20c927635/solidity-original`. Accessed 31 Mar 2018. [Cited on page 57.]

[249] Sovrin. `https://sovrin.org/`. Accessed 31 Mar 2018. [Cited on page 129.]

[250] "Global Legal Research Directorate Staff". Regulation of bitcoin in selected jurisdictions. *The Law Library of Congress, Global Legal Research Center*, 2014. [Cited on page 23.]

[251] Conor Svenson. Blockchain: Using cryptocurrency with java. *Java Magazine, January/February*, pages 36–46, 2017. [Cited on page 172.]

[252] Sweetbridge Whitepaper. `https://sweetbridge.com/public/docs/Sweetbridge-Whitepaper.pdf`. Accessed 31 Mar 2018. [Cited on page 136.]

[253] Szabo, Nick. Bit Gold. `http://unenumerated.blogspot.it/2005/12/bit-gold.html`. Accessed 31 Mar 2018. [Cited on page 23.]

[254] Block Chain Info Tags. `https://blockchain.info/tags`. [Cited on pages 66 and 110.]

[255] Yu Takabatake, Daisuke Kotani, and Yasuo Okabe. An anonymous distributed electronic voting system using zerocoin. 2016. [Cited on page 122.]

[256] Pavel Tarasov and Hitesh Tewari. The future of e-voting. *IADIS International Journal on Computer Science & Information Systems*, 12(2), 2017. [Cited on page 122.]

[257] Sasu Tarkoma. *Overlay Networks: Toward Information Networking.* CRC Press, 2010. [Cited on page 208.]

[258] The Economist, The trust machine. `https://www.economist.com/news/leaders/21677198-technology-behind-bitcoin-could-transform-how-economy-works-trust-machine`. Accessed 31 Mar 2018. [Cited on page 1.]

[259] The Path to Self-Sovereign Identity. `http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html`. Accessed 31 Mar 2018. [Cited on pages 127 and 128.]

[260] TIVI. `https://tivi.io/`. Accessed 31 Mar 2018. [Cited on page 122.]

[261] Andrew Tobin and Drummond Reed. The inevitable rise of self-sovereign identity. *The Sovrin Foundation*, 2016. [Cited on pages 127 and 129.]

[262] Top Trends in the Gartner Hype Cycle for Emerging Technologies 2017. https://www.gartner.com/smarterwithgartner/top-trends-in-the-gartner-hype-cycle-for-emerging-technologies-2017/. Accessed 31 Mar 2018. [Cited on page 3.]

[263] Kentaroh Toyoda, P Takis Mathiopoulos, Iwao Sasase, and Tomoaki Ohtsuki. A novel blockchain-based product ownership management system (poms) for anti-counterfeits in the post supply chain. *IEEE Access*, 2017. [Cited on page 137.]

[264] Alexander H Trechsel, Vasyl V Kucherenko, and Frederico Silva. Potential and challenges of e-voting in the european union. Technical report, 2016. [Cited on page 119.]

[265] Ujo Music. https://ujomusic.com/. Accessed 31 Mar 2018. [Cited on page 132.]

[266] Sarah Underwood. Blockchain beyond bitcoin. *Communications of the ACM*, 59(11):15–17, 2016. [Cited on page 3.]

[267] uPort. http://developer.uport.me/. Accessed 31 Mar 2018. [Cited on page 129.]

[268] uPort Whitepaper. http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf. Accessed 31 Mar 2018. [Cited on page 129.]

[269] Ferraiolo Vincent C. Hu, David, Kuhn Rick, Schnitzer Adam, Miller Sandlin, K. Robert, and Scarfone Karen. Guide to attribute based access control (ABAC) definition and considerations, 2014. [Cited on page 140.]

[270] Votem. https://votem.com/. Accessed 31 Mar 2018. [Cited on page 122.]

[271] VoteWatcher. http://votewatcher.com/. Accessed 31 Mar 2018. [Cited on page 122.]

[272] Vyper GitHub Repository. https://github.com/ethereum/vyper. Accessed 31 Mar 2018. [Cited on page 57.]

[273] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442, 1998. [Cited on page 72.]

[274] WikiLeaks: Banking Blockade and Donations Campaign. https://wikileaks.org/IMG/pdf/WikiLeaks-Banking-Blockade-Information-Pack.pdf. Accessed 31 Mar 2018. [Cited on page 21.]

[275] David Woo. Bitcoin: a first assessment. 2013. [Cited on page 23.]

[276] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, 151, 2014. [Cited on pages 55, 56, 185, and 191.]

[277] Aaron Wright and Primavera De Filippi. Decentralized blockchain technology and the rise of lex cryptographia. 2015. [Cited on page 2.]

[278] WSBI, Virtual currencies: passion, prospects and challenges. `https://www.wsbi-esbg.org/SiteCollectionDocuments/Virtual%20currencies_passion,%20prospects%20and%20challenges.pdf`. Accessed 31 Mar 2018. [Cited on page 23.]

[279] Haoyan Wu, Zhijie Li, Brian King, Zina Ben Miled, John Wassick, and Jeffrey Tazelaar. A distributed ledger for supply chain physical distribution visibility. *Information*, 8(4):137, 2017. [Cited on page 137.]

[280] James Wyke. The zeroaccess botnet–mining and fraud for massive financial gain. *Sophos Technical Paper*, 2012. [Cited on page 34.]

[281] Jesse Yli-Huumo, Deokyoon Ko, Sujin Choi, Sooyong Park, and Kari Smolander. Where is current research on blockchain technology?a systematic review. *PloS one*, 11(10):e0163477, 2016. [Cited on page 2.]

[282] Xiao Yue, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control. *Journal of medical systems*, 40(10):218, 2016. [Cited on page 123.]

[283] Filip Zagórski, Richard T Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L Vora. Remotegrity: Design and use of an end-to-end verifiable remote voting system. In *International Conference on Applied Cryptography and Network Security*, pages 441–457. Springer, 2013. [Cited on page 122.]

[284] Peng Zhang, Michael A Walker, Jules White, Douglas C Schmidt, and Gunther Lenz. Metrics for assessing blockchain-based healthcare decentralized apps. In *e-Health Networking, Applications and Services (Healthcom), 2017 IEEE 19th International Conference on*, pages 1–4. IEEE, 2017. [Cited on page 125.]

[285] Zhichao Zhao and T-H Hubert Chan. How to vote privately using bitcoin. In *International Conference on Information and Communications Security*, pages 82–96. Springer, 2015. [Cited on page 122.]

[286] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015. [Cited on pages 124 and 141.]

[287] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *Security and Privacy Workshops (SPW), 2015 IEEE*, pages 180–184. IEEE, 2015. [Cited on page 149.]