MASTER THESIS

Vivisecting Blockchain P2P Networks

A thesis submitted in partial fulfilment of the requirements
for the degree of Master in Civil Computer Science Engineering
Professional focus on Computer Systems and Security

in the

University of Liege
Faculty of Applied Science

Author:
Sami BEN MARIEM

Supervisor:
Dr. Benoit DONNET

Academic year 2018 - 2019

UNIVERSITY OF LIEGE

Abstract

University of Liege
Faculty of Applied Science

Master in Civil Computer Science Engineering
Professional focus on Computer Systems and Security

Vivisecting Blockchain P2P Networks

by Sami BEN MARIEM

"**A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution**"
(Satoshi Nakamoto, 2009, p. 1)

The idea behind this statement has been the key motivation for the development of the *"crypto-currencies"*. Indeed, those digital currencies rely on a recent implementation of an immutable and distributed ledger -i.e, the *Blockchain* - to allow transactions to take place in a distributed and decentralised manner without the need for any central authority. Blockchains are typically managed by peer-to-peer networks, which provide the support and substrate to the so-called *distributed ledger*, a replicated, shared and synchronised data structure, geographically spread across multiple nodes. Indeed, peer-to-peer networks allow the system to disseminate information among its peers while keeping it as much decentralised as possible.

In this paper, the network side of the blockchain technology will be studied, by characterising its topology and main properties from a purely network measurements-based approach. This will be done by analysing the most relevant cryptocurrency network : the *Bitcoin peer-to-peer network*.
First, the Blockchain technology as well as one of its most famous implementation -i.e., the Bitcoin - will be presented from a theoretical point of view, using well-known notions of Cryptography and Distributed Systems. Then, the methodology used for characterising the entities of the bitcoin network as well a passive measurements-based approach to unveil the topology of blockchain P2P network will be described. Finally, a characterisation of the bitcoin entities will be given through the combined analysis of multiple snapshots of the Bitcoin network as well as by using other publicly available data sources. As it is shown and discuss, many key ideas and methods are likely to be reusable in various other fields using the blockchain technology. Therefore, the impact of this thesis reaches far beyond the Bitcoin technology itself.

Among other relevant findings, it is shown that *(i)* the size of the BTC network has remained almost constant during the last 12 months – since the major BTC price drop in early 2018, *(ii)* most of the BTC P2P network resides in US and EU countries, and *(iii)* despite this western network locality, most of the mining activity and corresponding revenue is controlled by major mining pools located in China.

Remark: Several results that are presented in this thesis have been previously presented in the paper : *"Vivisecting Blockchain P2P Networks: Unveiling the Bitcoin IP Network"*

Acknowledgements

First of all, I would like to thank Professor Benoit Donnet, my research supervisor, who has given me the opportunity to broaden my knowledge and skills through numerous successful projects. Amongst them, my internship at the Austrian Institute of Technology. This experience gave me insight into the field of Network Measurements and helped me open my mind to unique sensations. His guidance and his caring attitude towards my work have greatly motivated my determination to succeed in my academic carreer.

Along with Professor Donnet, I would like to express my sincere gratitude to Dr Pedro Casas, my internship supervisor, who provided invaluable help, through discussions and recommendations, for the realisation of my internship. He has been of great aid and has demonstrated fraternal commitment towards my objectives. Dr. Casas also introduced me to the wonderful city of Vienna and some of its particularly friendly inhabitants.

I would like to extend my thanks to the teaching staff of the Faculty of Applied Science for the devotion and kindness showed during my journey as a student.

Finally, I wish to express my gratitude to my family and my friends, for their support and encouragement throughout my studies.

Contents

List of Figures

List of Tables

List of Abbreviations

FLP	**F**undamental **I**mpossibility **Results**
DLT	**D**istributed **L**edger **T**echnology **BTC**
Bitcoin	
BNI	**B**itcoin **N**ode **I**ndex
WAN	**W**ide **A**rea **N**etwork
TCP	**T**ransmission **C**ontrol **Protocol**
UDP	**U**ser **D**atagram **P**rotocol
IP	**I**nternet **P**rotocol **RTT**
Round **T**rip **T**ime	
SHA	**S**ecure **H**ash **A**lgorithm
DNS	**D**omain **N**ame **S**ystem **API**
Application **P**rogramming **I**nterface	
ARPANET	**A**dvanced **R**esearch **P**rojects **A**gency **Net**work **DDoS**
Distributed **D**enial of **Service**	
P2P	**P**eer-**to**-**Peer**

Chapter 1

Introduction

The modern computer era began with the invention of the Turing Machine by Alan Turing in 1936 [107]. Since then and up to the invention of computer networks, computer systems operated independently. Indeed, technology lacked a means to make them communicate with each others.
In 1969, the *U.S. Department of Defense* launched the first nodes of the **A**dvanced **R**esearch **P**rojects **A**gency **Net**work (ARPANet) [34], known as the precursors of the first and most famous **W**ide **A**rea **N**etwork: the *Internet*. With the competing development of intercontinental missiles prototypes, the *Department of Defense* was concerned about their ability to ensure communication in the event of a nuclear strike. Therefore, *Paul Baran*, a pioneer in the development of computer networks, decided to address this topîc. He concluded that the strongest communication system would be a distributed network of computers having the following properties: *(1)* sufficient redundancy to avoid the failure of a subset of links or nodes to isolate any of the correct nodes, *(2)* communications being done through signals traversing a series of nodes from source applications to destination each one routing the signal towards the destination [34]. This has laid the foundations for current computer networks and, by extension, for distributed systems.

From that moment on, the distributed computing field witnessed the rise of a wide variety of applications trying to solve problems in a cooperative and distributed manner (e.g., BitTorrent, Gnutella, Napster. Indeed, distributed applications offer several advantages compared to standard approaches (cfr. *Section 2.1.1*).
The financial sector was no exception to this rule and the idea of a fully distributed digital currency had been around since the early 1980s. The first attempts to build digital currencies required a central authority (e.g., financial institutions) to coordinate the exchange of money and check for fraud [27, 71, 71]. However, the vision of a digital currency not needing any coordinating entity was already present and approaches like the ones of B-Money [32], bit gold [102] and RPOW [48] were already interpreting the solution of a cryptographic puzzle -i.e., a proof of work - as something valuable.

In 2009, the idea of a fully decentralised and distributed virtual currency was finally put into practice with the deployment of the *Bitcoin* (BTC). The design of the technology, combining decades of research [14, 73, 76], had been announced the year before in a white paper published by Satoshi Nakamoto [81]. It is relying on the implementation of the distributed ledger: the *Blockchain* to record transactions in a distributed and decentralised fashion. Then, any entity in the system is able to verify and audit transactions inextensively without the need of any trusted third-party.

Similarly to Gnutella [20], the *Bitcoin* blockchain is built on the top of a decentralised peer-to-peer (P2P) network, used to propagate relevant information such as transactions between entities. As explained in [33], the security of the blockchain technology is highly dependent on the security of its underlying P2P Networks. Therefore, a need for characterising those networks became crucial.

1.1 Related Works & Contributions

Previous papers have studied the BTC blockchain, mainly in terms of executed transactions, through the analysis of the publicly available distributed ledger. For example, [57, 75] focused on the BTC

transactions as observed at the BTC DLT, [104] studied the security of the BTC P2P network, [22] analyses the temporal generation of BTC blocks, [74] focuses on the energy footprint of BTC mining, etc.

On the other hand, other papers also tried to study the P2P network topology and characteristics of BTC and other popular blockchains [65, 78, 84]. However, the continuous evolution of the DLT technology and the potential security issues linked to unveiled P2P topologies (e.g., Eclipse Attack) [33, 90, 100] result in constant updates of the underlying protocols, making some of the previous proposals no longer applicable. In particular, back in 2015, Miller et al. [78] proposed a comprehensive technique to discover P2P links in the BTC network and identify topologically-influential nodes, relying on the analysis of the broadcast messages over the network. However, the proposed technique is no longer applicable to the current BTC protocol, which has been updated to remove relevant timing information used in [78].

Among other relevant findings, it is shown that *(i)* the size of the BTC network has remained almost constant during the last 12 months – since the major BTC price drop in early 2018, *(ii)* most of the BTC P2P network resides in US and EU countries, and *(iii)* despite this western network locality, most of the mining activity and corresponding revenue is controlled by major mining pools located in China.

1.2 Thesis Outline

This thesis presents a combination of the oldest and the newest techniques to unveil *Blockchain peer-to-peer networks*. First, *Chapter 2* describes and develops the relevant concepts related to the blockchain technology when looked at through the lens of the distributed system theory. From this point, some key characteristics of its most famous implementation: the *Bitcoin* will be depicted. Based on the notions defined in the previous chapter, *Chapter 3* details and motivates the approach chosen for characterising *Blockchain peer-to-peer networks*. Then, *Chapter 4* will deepen the knowledge and the visibility of the *Bitcoin peer-to-peer network* by providing some of the results obtained using the aforementioned approaches. Finally, *Chapter 5* conclude the work that has been presented and proposed some future works.

Chapter 2

State of the Art

In this chapter, an overview of the most relevant blockchain and Bitcoin concepts is presented as well as the theory on which they rely.

The chapter is structured as follows: First, *Section 2.1* and *Section 2.2* introduce the notions on which the blockchain relies when looked from a purely theoretical point of view. Then, *Section 2.3* describes the blockchain technology in more details, using the concepts defined in the previous sections. Finally, *Section 2.4* focus on depicting a famous implementation of the blockchain technology: the *Bitcoin*.

2.1 Background on Distributed Systems

2.1.1 Definition

As described in [24, 31, 98], a distributed system can be defined as a collection of **independent entities** which communicate through a communication medium to achieve a common goal and that appear as a **single coherent system** to its users.

Entities

The entities involved in distributed systems are commonly referred to as **nodes**, **agents** or **processes**. Nodes can either be hardware devices or software processes and can act independently from each other. In practice, nodes are independent but programmed to achieve a common goal by communicating with each others through a communication medium.

The less restrictive assumption on nodes ensures that each of those entities is autonomous, programmable, asynchronous and failure-prone (cfr. *Section 2.1.2*).

Communication Medium

The communication medium involved in distributed systems may be any kind of network. However, practice shows that a distributed system is often organised as an overlay network (e.g., peer-to-peer networks) [108].
In principle, the network used as a communication medium must be connected to allow each peer to route a message to any other through the network.

The less restrictive assumption on communication links being the unreliable communication medium (cfr. *Section 2.1.2*).

Distributed Systems vs Non-Distributed Systems

Compared to non-distributed systems, distributed systems provide some advantages such as:

- **Scalability**: Single coherent systems can be expanded by increasing hardware performance -i.e., scaling vertically - of the system. However, this may not be possible or profitable after a while.

In opposition to that, distributed systems can be expanded by adding more machines to the system -i.e., scaling horizontally - to handle the need for more performance.

- **Availability & Reliability**: Distributed systems can avoid the single point of failure problem. Indeed, the system can be designed so that several machines can provide the same services/data in case of failure. Indeed, replication of services/data is part of the solution to provide fault-tolerance.

- **Low Latency**: Distributed systems also help reducing the latency of queries through replication. Indeed, the speed at which a network packet travels the world is physically bounded by the speed of light. Therefore, there is a speed limit at which a peer can communicate with another that is located far away. The only variable left is the location of the nodes answering peer requests.
  Distributed systems allow a user to query the closest node that can provide the needed service/data and decrease the delay before the answer is received.

On the other hand, distributed systems are very difficult to build and are known to bring a lot of challenges such as communications (e.g., how to provide reliable networking with unreliable communication medium), concurrency (e.g., how to handle access to shared resources), consistency (e.g., how to ensure that all peers in the system have up-to-date information) and fault-tolerance (e.g., how to make your system operate under malicious behaviour/nodes failure) [24, 31].

2.1.2 Distributed Systems Model

The core of any distributed system is a set of distributed algorithms [24] (e.g., failure detection, leader election) that are run by the different entities of the system. However, those algorithms must rely on some assumptions related to the nodes/links behaviour to be correct (e.g., synchronous nodes). Indeed, one of the consequence of each node's independence and autonomy is that each node may have its own notion of time and may crash/behave differently.

A distributed system model defines the assumptions on which the distributed algorithms rely to be correct. These assumptions define the failure and timing behaviour of nodes and channels in the system. It consists of a combination of three categories of abstraction [24]:

- **Process Abstractions**: Define the possible behaviour of the different entities involved in the system. The following process abstractions are possible:

  - **Crash-stop** Failure: An entity of the system may stop taking part in the system at some point and will never recover.
  - **Omissions** Failure: An entity of the system may omit sending/receiving a message that should have been sent/received (e.g., due to Buffer overflow).
  - **Crash-recovery** Failure: An entity of the system may stop taking part in the system but may at some point recover.
  - **Byzantine/arbitrary** Failure: An entity of the system may behave arbitrarily (e.g., a malicious nodes).

- **Link Abstractions**: Define the possible behaviour of the links used as a communication medium between the entities of the system. The following link abstractions are possible:

  - **Fair-Loss** Links: The channel will deliver any message sent with a non-zero probability (e.g., UDP).
  - **Stubborn** Links: The channel will deliver any message sent infinitely many times.
  - **Perfect** Links: The channel will deliver any message sent exactly once. (e.g., TCP).

- **Timing Abstractions**: Define the possible behaviour of the entities and the links with respect to the passage of time. The following time abstractions are possible:
  - **Synchronous** System: Known upper bound on process computation and message transmission delays in the systems. For example, processes have access to a local physical clock with a known upper bound on clock drift and clock skew.
    This assumption helps inducing information from the absence of activity of an agent.
  - **Partially Synchronous** System: The system is synchronous most of the time but there are periods where the synchronous system assumption does not hold.
  - **Asynchronous** System: There are no timing assumptions on processes and links. This mainly implies that it is impossible to tell if a processor has failed, if a message has been lost, or if a longer time is needed for the message to arrive.

As illustrated in *Figure 2.1*, it is obvious that some assumptions (e.g., crash-stop, synchronous systems) are stronger than others (e.g., byzantine, asynchronous systems) which causes their applicability to be limited.

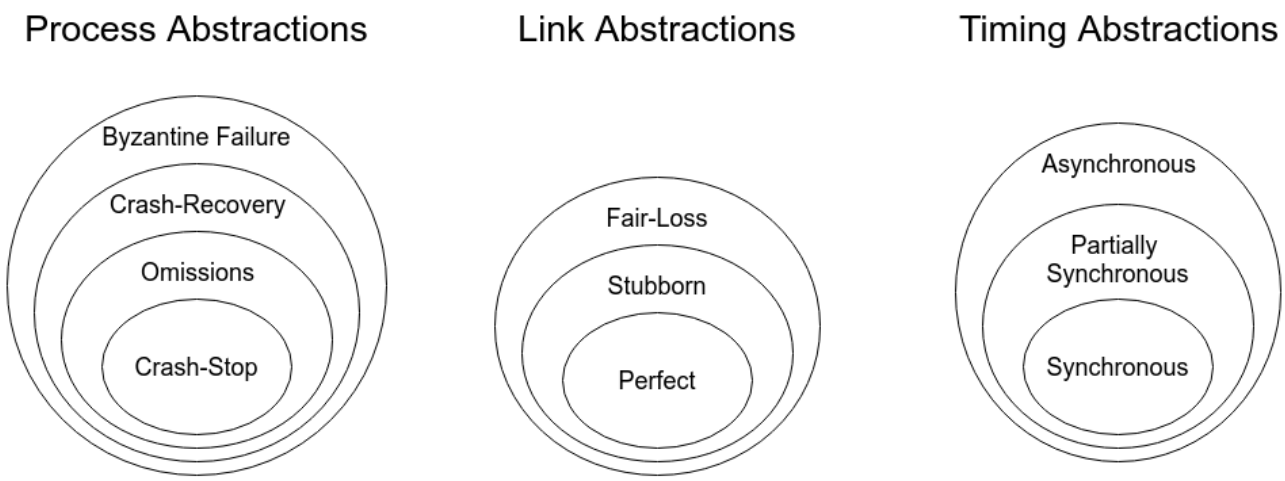


FIGURE 2.1: Abstractions Hierarchy

2.1.3 Consensus

A fundamental problem in distributed computing and multi-agent systems is to make processes **agree** on some data value [24, 31]. This problem, known as the **consensus** problem is key to solve many other problems such as failure detection, group membership management, Atomic Broadcast [26, 56].

A distributed algorithm relying on a distributed system model (assumptions on nodes/links behaviour) solves the consensus problem if and only if it respects the following properties [24, 31]:

1. **Termination**: All non-faulty processes eventually decide on a value.
2. **Agreement**: All processes that decide do so on the same value.
3. **Validity**: The value that has been decided must have been proposed by some process.

The *Fundamental Impossibility Result (FLP)* states that all three properties cannot be guaranteed in an asynchronous distributed system with one faulty process without making additional assumptions on the system behaviour [49]. A protocol guaranteeing consensus among *"n"* processes with no more than *"t"* processes that failed is said to be **t-resilient**.

Byzantine Generals Problem

The *Byzantine Generals Problem* is an abstract expression of the consensus problem in an **asynchronous**, **byzantine fault-tolerant** system using a **fair-loss link**.

Solving the consensus problem in such a system is one of the most difficult problems in distributed computing. Indeed, only the weakest assumptions possible are being done regarding processes and links behaviour.

As a consequence of the *FLP Impossibility*, this problem has been proved to be unsolvable without making additional assumptions on the system behaviour [70].

2.1.4 CAP Theorem

The *CAP theorem*, also named *Brewer's theorem* has proved that a distributed system cannot achieve simultaneously [51, 55]:

- **Consistency**: Property ensuring that all nodes in a distributed system have a same copy of the latest version of the data (Every read returns the most recent write or an error).
- **Availability**: Property ensuring that the system is accessible for use and that it answers incoming requests without any failures (Every request returns a (non-error) response without any guarantee about consistency).
- **Partition Tolerance**: Property ensuring that the system continues to operate uphold consistency and/or availability despite any (group of) node/link failure.

However, no distributed system is safe from network failures which generally cause partitioning to be tolerated. Therefore, this theorem mainly implies that in the presence of a partition, one has to choose between consistency and availability.
When choosing consistency over availability, the system will return an error or a time-out if a particular information cannot be guaranteed to be up to date due to network partitioning. In opposition to that, when choosing availability over consistency, the system will always process the query and try to return the most recent available version of the information, even if it cannot guarantee it is up-to-date due to network partitioning. In the absence of partitioning, the properties of availability and consistency can both be guaranteed.

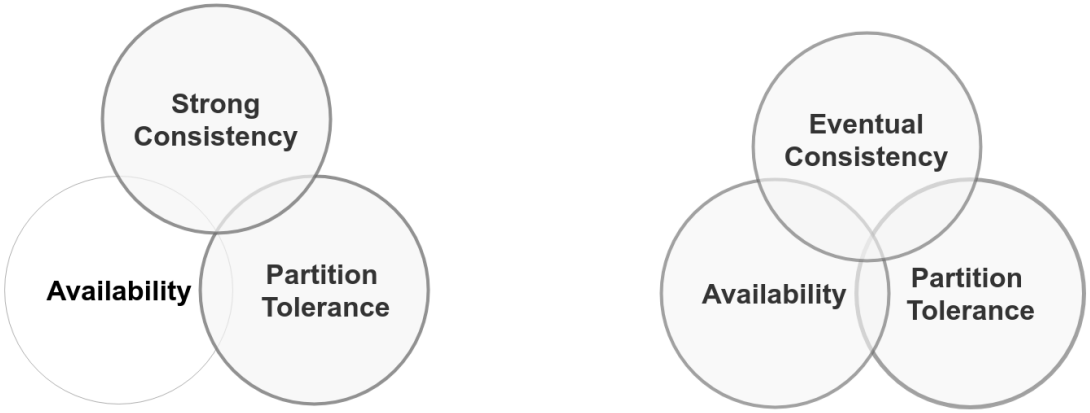
Strong vs Eventual Consistency

As mentioned in *Section 2.1.1*, distributed systems partially achieve availability through the use of replicas. Indeed, several nodes are able to provide similar services or pieces of data. However, the entire system needs to have a same version of the replicas to reach consistency. Therefore, nodes need to agree on the shared version of the replicas through a consensus mechanism.

In practice, conflicts may arise among the nodes when choosing the version of the replicas. During the period of conflict, two behaviour of the system can be expected in case of request: *(1)* Return the latest version that has been decided or, *(2)* Return an arbitrary value depending on the node being request.

In the first case, distributed system theory outlines that *Strong Consistency* is the property of a distributed system ensuring that all nodes need to agree on a version of the replicas before making them available. Therefore, updates are considered to be done by all nodes of the system simultaneously and any read return the same value no matter which peer has been queried.

The second behaviour is often referred to as *eventual consistency* [42]. Indeed, the system may return stale data during the consensus period but conflicts will eventually converge towards a same decision. The system will be considered as *inconsistent* during some period but it always converge toward a *consistent* state.



(A) CAP Theorem with Strong Consistency (B) CAP Theorem with Eventual Consistency

FIGURE 2.2: CAP Theorem: Strong vs Eventual Consistency

In case of network failure leading to partitioning, consensus among all peers cannot be guaranteed. Thus, the *CAP Theorem* forces the system to choose between providing *Consistency* or *Availability*. As illustrated in *Figure 2.2a*, the strong consistency property is ensured at the cost of availability as the system may need to delay/to reject some of the queries to avoid returning stale data. In opposition to that, *Figure 2.2b* shows that the eventual consistency property allows a distributed system to be available at all time

2.1.5 Usages

Distributed Database

Distributed systems can typically be used to store data in a distributed fashion. Indeed, distributed databases consist of a collection of multiple, logically interrelated databases distributed over a computer network. They are managed by a distributed database management system that provides an access mechanism to the user so that the database distribution remains hidden [88].

Distributed databases partly rely on replication to handle fault-tolerance and to improve performance [116]. Indeed, replication of data over different nodes improves reliability and availability of data in case of node failure as the system is not relying on a single node to store a piece of data -i.e., no single point of failure. In addition to that, data replication also improves performance and scalability as it results in decreasing the response time of the system and the spreading of the queries among the different nodes.

However, handling replication in a distributed system is not a trivial problem. Indeed, to remain consistent, the system has to make sure that the effect of any update on a piece of data is reflected on each and every copy.

Distributed database systems may handle data replication over the nodes in several ways such as [29]:

- **Multi-master replication** architecture allows data to be stored by a group of nodes, and updated by any member of the group. All members are responsive to client data queries.
  The multi-master replication process is responsible for propagating the data modifications made

by each member to the rest of the group, and resolving any conflicts that might arise between concurrent changes made by different members (consensus problem).

- **Master-Slave Replication** architecture elects a single node of the system as the *master* for a given piece of data. This *master* will be the only node allowed to modify that data. Therefore, any other node needing to modify the piece of data first have to ask the *master* which will decide whether or not the modification is applied.

Distributed Ledger

A distributed ledger [89] is a particular type of distributed database providing a consistent, immutable, append-only database that is replicated, synchronised and shared across a distributed network.

It may be centralised (e.g., central entity giving permission to append) or decentralised (e.g., permission given through network consensus). It is important to mention that it can hold any kind of data even though it has mainly been famous for the record of transactions (cfr. *Section 2.4*).

2.2 Background on Cryptography

Cryptography refers to a field of cryptology that consists of a set of principles, methods and techniques to ensure the data encryption and decryption [39]. Regarding information security, cryptographic tools can provide several guarantees on encrypted data such as [64]:

- **Confidentiality**: *"The property that information is not made available or disclosed to unauthorised individuals, entities, or processes."*[1]
- **Integrity**: *"The property that data has not been altered or destroyed in an unauthorised manner."*[1]
- **Authenticity**: The property of data that it has an identified origin[2].
- **Non-repudiation**: The property of data that it has an identified origin that cannot deny its previous commitment, actions, ...

2.2.1 Symmetric Cryptography

Symmetric cryptography refers to cryptographic algorithms using a same secret cryptographic key both for encryption of plain texts and decryption of encrypted texts [64, 97].

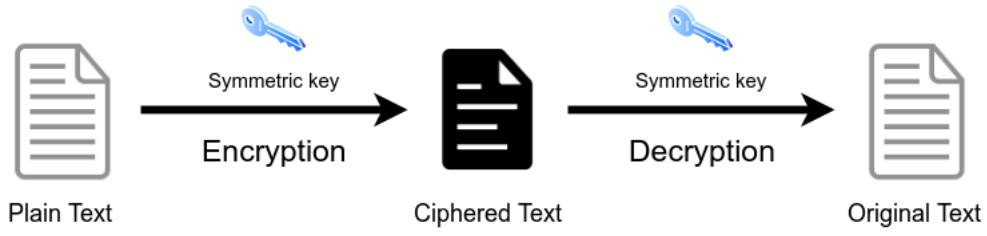


FIGURE 2.3: Symmetric Encryption

Therefore, the protagonists need first to exchange the key that will be used for encryption/decryption. Ideally, the encryption/decryption algorithm is not sensitive to any attack and the encryption safety only depends on the non-disclosure of the key.

[1] ISO/IEC PDTR 13335-1 (withdrawn standards)
[2] "Origin" may refer to several concepts such as date of origin, entity that created the piece of data, ...

2.2.2 Asymmetric Cryptography

Asymmetric cryptography refers to cryptographic algorithms using a pair of keys each being used either for encryption or decryption [64, 97]. In practice, a stakeholder will own a **public key** and a **private key**, each used for either for decryption or for encryption of data.

In the scenario illustrated in *Figure* 2.4, the encryption aims at ensuring that the encrypted data will only be read by the allowed users -i.e., **confidentiality**. Indeed, once the piece of data has been encrypted with a public key, only the owner of the corresponding private key will be able to decrypt it.

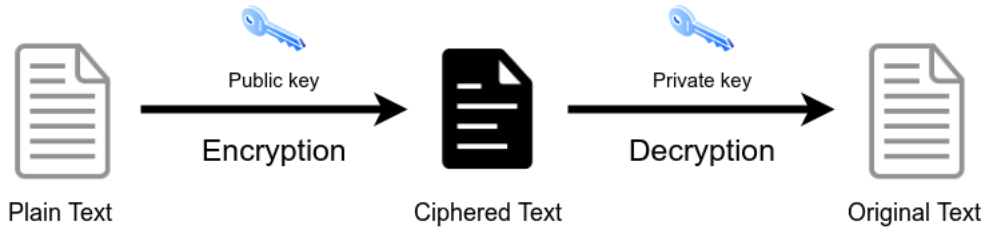


FIGURE 2.4: Asymmetric Encryption with public key

On the other hand, *Figure* 2.5 shows a scenario where the encryption aims at digitally signing a piece of data. Indeed, once the piece of data has been encrypted with a private key, only the corresponding public key will be able to decrypt it. Therefore, anybody receiving the encrypted piece of data will be sure that it has been signed by the owner of the private key.

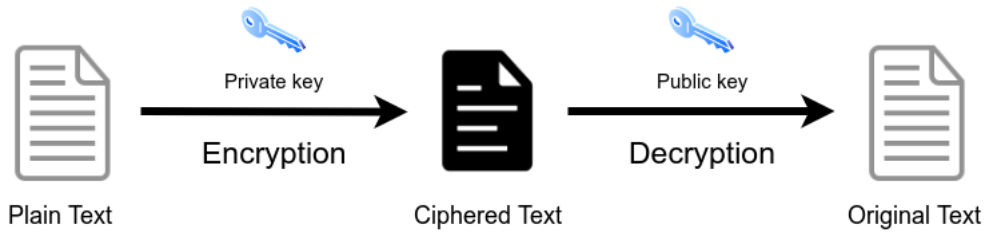


FIGURE 2.5: Asymmetric Encryption with private key

Ideally, the encryption/decryption algorithm is not sensitive to any attack and the encryption safety only depends on the non-disclosure of the private key.

2.2.3 Cryptographic Hashing Functions & Data Structure

A **hash function** is any function h that maps an input x of arbitrary finite length to an output $h(x)$ of fixed finite length n, called a hash [64, 97].

A **cryptographic hash function** is a hash function which ideally respects the following properties [39, 95]:

- **Pre-image resistance**: Given a hash y, it requires 2^n to find an input x such as $h(x) = y$.[3]
- **Second Pre-image resistance**: Given an input x, it requires 2^n to find another input x' such that $h(x) = h(x')$.[3]

[3]There is no approach more efficient than the brute-force approach.

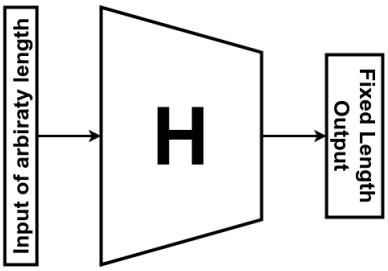


FIGURE 2.6: Hash Functions

- **Collision resistance**: It requires $2^{n/2}$ hash calculations to find two inputs x and x' such that $h(x) = h(x')$.[3]

Cryptographic hash functions are used to ensure data integrity (e.g., data hashes), authenticity and non-repudiation (e.g., digital signatures) [39, 95].

2.2.4 Usages

Data Integrity

Data hashes may also be used to ensure data integrity.
In practice, the hash-value corresponding to a particular piece of data is computed and then protected in some manner. At a subsequent point in time, an entity wanting to check data integrity will recompute the hash-value of the piece of data and will compare the computed hash-value with the original to check if the piece of data has been altered.

Merkle Trees

Merkle Trees [16] aims at ensuring data integrity.
Indeed, a merkle tree or a hash tree is a tree data structure where every leaf node is labelled with the hash of a piece of data contained in a bigger data structure. Then, every non-leaf nodes are labelled with the cryptographic hash of the labels of their child nodes.
Merkle trees allow efficient and secure verification of the integrity of a data structure.

Example Consider a data structure consisting of *"n"* piece of data {l1, l2, ... , ln} sent over an unreliable channel.
The sender first computes the merkle tree of that data structure and then sends its root along with the data structure over the unreliable channel. In order for the receiver to check the integrity of the data structure it has received, it simply has to compute the merkle tree and compare its root to the one received.

In addition to that, merkle trees allow fast-tracking of *corrupted* pieces of data. Indeed, if the data structure has been found to be corrupted, the receiver only has to request the hashes of the subtrees until the corrupted pieces of data are spotted.
In practice, demonstrating that a leaf node is a part of a given binary hash tree only requires computing a number of hashes proportional to the logarithm of the number of leaf nodes of the tree.
Therefore, spotting the corrupted piece of data only consist of querying, computing and comparing **log(n)** hashes.

This mechanism has several advantages:

- **Fast Integrity Check**: Checking the integrity of a piece of data only requires to compute hashes.

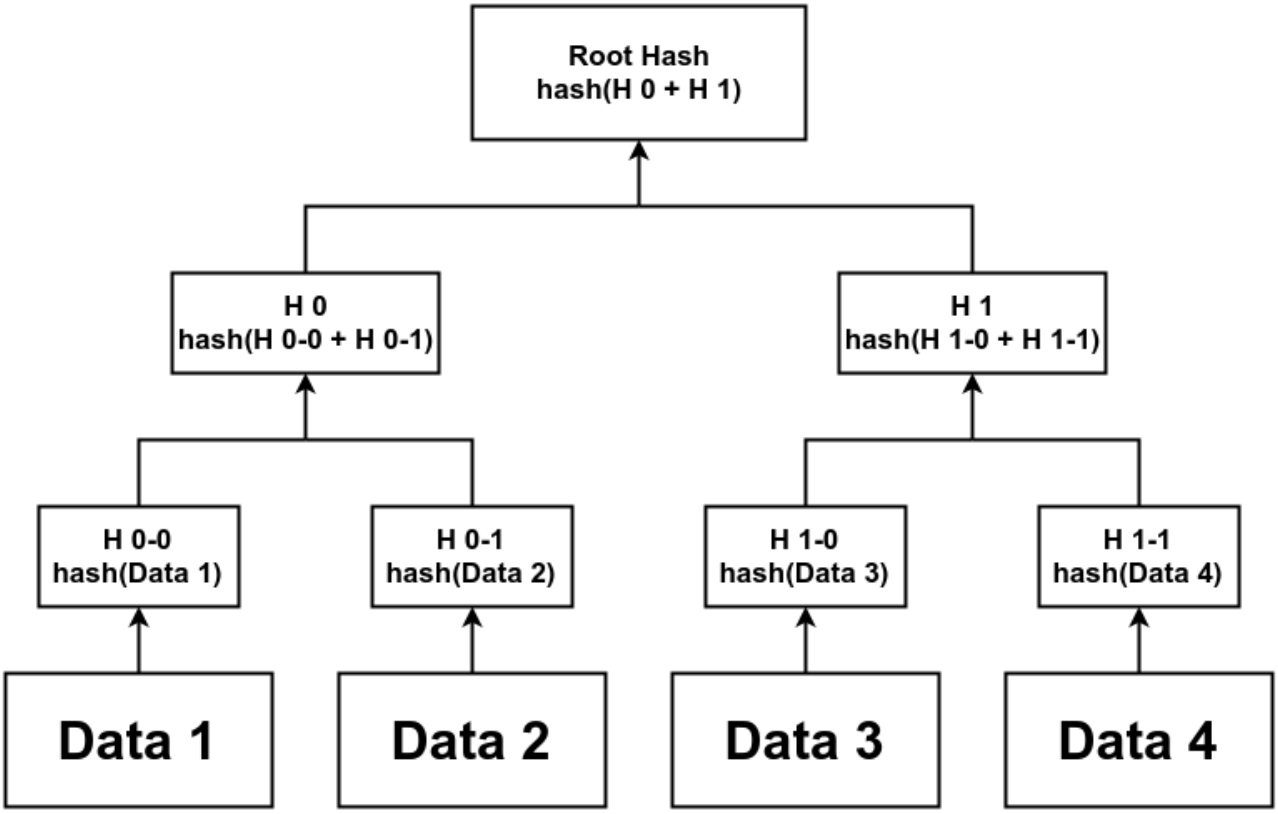


FIGURE 2.7: Merkle Tree

- **Fast Identification of Corrupted Data**: Spotting corrupted pieces of data does not need the entire data structure to be compared to the original. Therefore, it allows to only re-send the corrupted pieces of data.

Digital Signatures

Digital signatures aim at ensuring data authenticity and non-repudiation.
They can be created both using symmetric and asymmetric cryptography. Indeed, a signer signs a piece of data either by:

- Encrypting the data using a previously shared secret key. Therefore, authenticity mainly relies on the process of sharing the secret key and keeping it secret or,
- Encrypting the data using its own private key. Another entity knows that the piece of data was signed by the signer as it will only be able to decrypt the piece of data using the signer's public key. Therefore, authenticity mainly relies on the process of sharing the public key.

In practice, a piece of data is usually hashed and it is the hash-value that is signed by the signer. Then, an entity wanting to check the authenticity of the data will compute its hash and will compare it to the previously decrypted original hash (using either the previously shared secret key or the signer's public key). This mechanism saves both time and space compared to signing the message directly (message size is usually bigger than the size of a hash). *Figure 2.8* illustrates this mechanism.

Here, the inability to find two inputs with the same hash-value is ensuring that another entity will not be able to claim something that another entity has signed (data authenticity) and that the signer will not be able to deny signing the data (nonrepudiation).

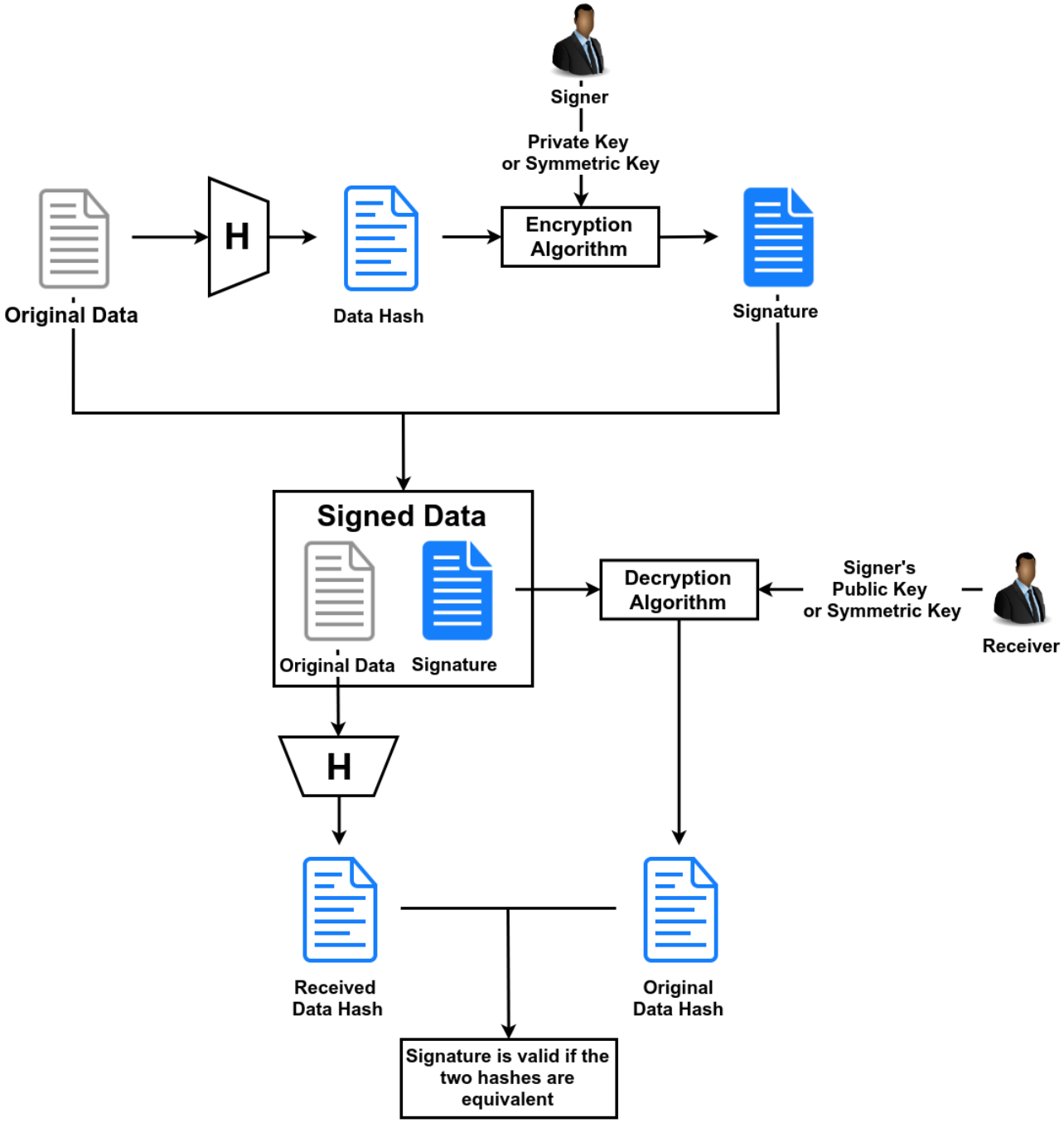


FIGURE 2.8: Digital Signature Process

2.3 Background on Blockchain

2.3.1 General Overview

The Blockchain technology is an implementation of the distributed ledger technology [15] relying on three main components:

- A **cryptographic hash function**: The system needs a mechanism for linking a block to the previous (e.g., SHA-256). This mechanism ensures that the modification of a block included in the blockchain results in an incoherence regarding the linkage between blocks and thus an invalidation of the blockchain.

- A **distributed consensus mechanism**: The system needs a consensus mechanism (e.g., Proof of Work) that allows its nodes to agree on a common ledger -i.e., a common chain of blocks - and thus, to have a consistent database among the network.
- A **communication medium**: The system needs a medium through which the different nodes communicate and exchange information (e.g., a peer-to-peer network).

Altogether, those assumptions ensure that any record added to the blockchain cannot be altered retroactively, without the alteration of all subsequent blocks and the consensus of the system. In practice, the difficulty of the process is ensuring the *immutability* of data added to the blockchain.

In the following sections, the words *blockchain*, *distributed ledger*, and *ledger* will be used interchangeably as well as the words *peers*, *nodes* and *agents* which refer to the entities involved in the system.
In addition to that, it is of crucial importance not to confuse *nodes* with *users* who use the system without necessarily being part of it.

2.3.2 Definition & Architecture

A blockchain is made of a continuously growing list of records: the blocks. As illustrated in *Figure 2.9*, a block consists of a *block header* and a *block body* [115] which are structured as follow:

- The block header typically contains:
  - A **cryptographic hash of the previous block**: Protection against changes in blocks that have already been accepted in the chain[4].
  - A **timestamp**: Protection against re-usage of a block.
  - A **Merkle Tree's root hash**: Protection against changes happening during the transmission of the block over unreliable communication medium.
- The block body is mainly composed of an indicator on the amount of data that is included in the block as well as the data itself. The maximum size allowed for a block must be defined in the implementation of the blockchain.

All in all, blockchains allow data to be permanently added, timestamped, verified, and shared securely among nodes. Therefore, it can be seen both as a secured timestamp server and as a secured distributed database.

A **valid** blockchain is a blockchain in which for all blocks, the block's hash corresponds to its child's parent hash.

2.3.3 Taxonomy

Just like other distributed systems involving storage of data, there are several levels of access to the distributed ledger [53]:

- Reading data included in the ledger, perhaps with further restrictions (e.g., user can only read data they submitted).
- Submit data for inclusion in the ledger.
- Inclusion of data in the ledger (participate to the consensus process).

Therefore, blockchains can be classified into several categories based on which peer has access to which level.

[4]The first block of a blockchain is called the *genesis block* and is the only exception for which this field is empty.

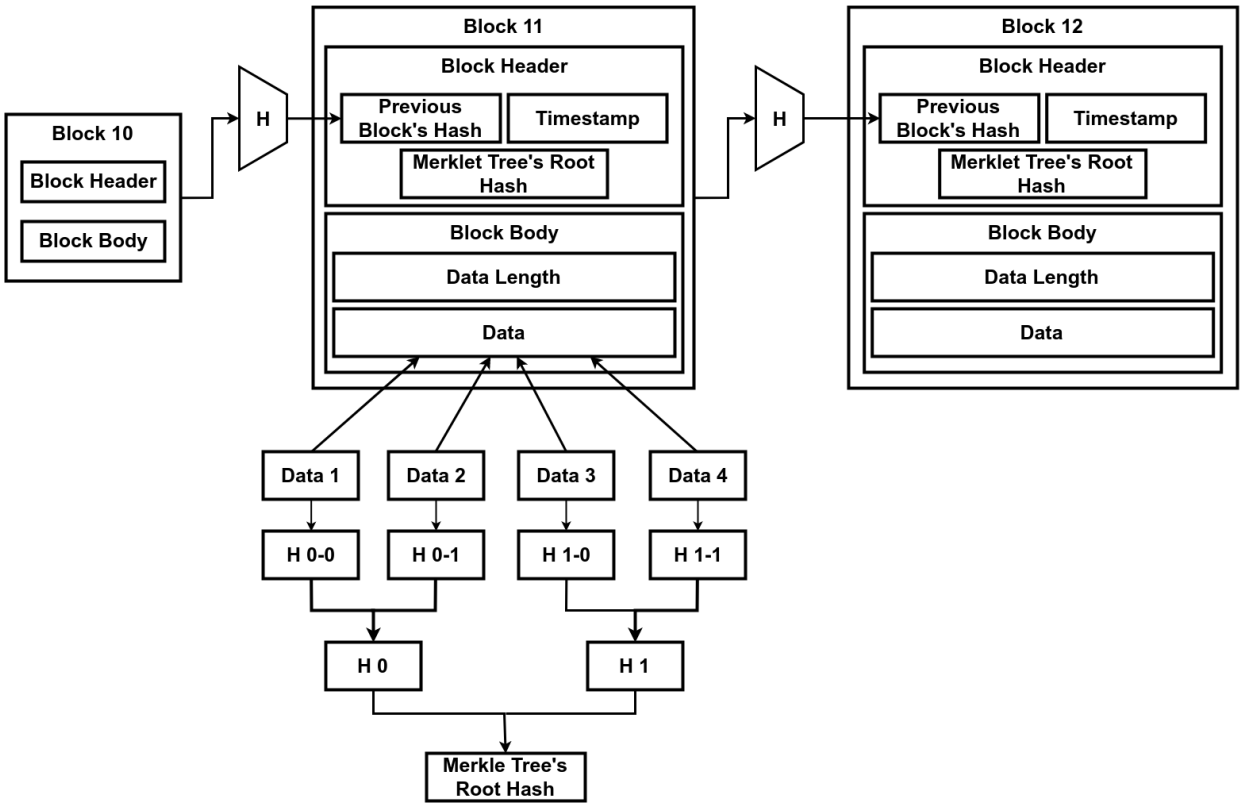


FIGURE 2.9: Blockchain Structure

Permissioned vs Permissionless

Blockchains can be classified with respect to the third level of access. Two kinds of blockchains can be distinguished [53]: *permissionless blockchains* which do not specify any restriction on the entities that can be involved in the consensus process (e.g., Bitcoin) and permissioned blockchains which restrict this set of entities to a predefined subset of actors (e.g., Ripple [61]).

Public vs Private

Blockchains can be categorised according to the first and second levels of access [115]: *public blockchains* that allow everyone to read data from the ledger and submit data for inclusion in the blockchain (e.g., Bitcoin , Ethereum) and *private blockchains* that only allow a pre-defined subset of nodes (e.g., coming from a specific organisation) to read/submit data of/to the blockchain (e.g., Hyperledger Fabric).

Centralised vs Decentralised

The notion of *decentralisation* is insidiously coming back within those categories. Indeed, it is clear that private, permissioned blockchains cannot be decentralised as only a set of nodes are involved in the consensus/submission process.
Although it is often the case, it is important to mention that the notion of decentralisation cannot be reduced to the consensus mechanism and that it also encompasses other notions (e.g. organisation of the underlying network).

Blockchains can be *centralised* (e.g., World Food Programme, IBM's HyperLedger Fabric) or *decentralised* (e.g., Bitcoin, Ethereum) mainly depending on the consensus mechanism they rely on but not only. A centralised blockchain only trusts a central authority [5] for making decisions in the system (e.g.,

[5] A central authority may consist of several nodes.

confirming the inclusion of data in the ledger) while a decentralised blockchain is not.

Byzantine Fault-Tolerant vs Others

Blockchains can be grouped according to their level of fault-tolerance. Indeed, the technology is said to be byzantine-fault tolerant (cfr. *Section 2.1.2*) when it does not make any assumption on the behaviour of the entities involved in the systems (e.g., Bitcoin, Ethereum). Therefore, the services provided by the distributed ledger must still be ensured in case of nodes acting maliciously. On the other hand, blockchains may consider some stronger assumptions regarding the behaviour of nodes.
It is obvious that for a blockchain system to be byzantine fault-tolerant, it must rely on a byzantine fault-tolerant consensus mechanism, network and cryptographic hash function[6].

2.3.4 Consensus Mechanisms

Problem Definition & Challenges

Similarly to any other distributed system, the distributed ledger also have to face the problem of consensus between its entities to ensure consistency of its database. In practice, the consensus mechanism is used for deciding which block is added to the blockchain.

In this section, only byzantine fault-tolerant and decentralised consensus mechanisms will be described. As mentioned in (cfr. *Section 2.1.2*), byzantine fault-tolerant systems may face malicious nodes -i.e., called **adversaries**.

Proof of Work

The **Proof of Work** [59, 115] is a consensus mechanism that has been inspired by protocols designed to prevent spam attacks and Distributed Denial of Service (DDoS) attacks [14].

Principle The main rules on which the mechanism relies to ensure consensus over the network is that the involved set of entities will always keep the *valid* blockchain that led to the highest estimated amount of computations, the so-called **proof of work**. Indeed, the amount of work will represent a measure of trust in the system. Therefore, a block will be added to the blockchain if it is the block that most increase the amount of work represented by this blockchain.

In practice, the different entities involved in the consensus process, the so-called **miners** will compete in solving a **cryptographic puzzle** for adding a block to the blockchain. This process is known as **mining**. The miners that succeed in mining a block will be rewarded with an incentive that must be defined in the implementation.

The puzzle being solved must have the following properties:

1. **Asymmetric**: Computing the solution to the puzzle is slow but checking if an answer is correct is fast.
2. **Guess-only**: Computing the solution can only be done via brute-force.
3. **Adaptive Difficulty**: The difficulty of the puzzle can be updated according to the computational power of the network. This mechanism allows to maintain a stable block generation rate with a varying network power.

[6]The notion of byzantine fault-tolerance may evolve with time (e.g. cryptographic functions are not byzantine fault-tolerance if considering quantum computers).

4. **Block Dependant**: The puzzle must depend on the block which is being *mined*. This ensures that once a block has been mined, the other miner will have to start back from the start of their research.

Once a solution to the puzzle is found by an entity, it will be added to the block and the block will be broadcast to the network. Then, the other nodes will confirm the correctness of the solution and append the block to their version of the blockchain.

Proof of Stake

The **Proof of Stake** is a consensus mechanism that relies on the stake that entities have in the system as a trust measure [111]. It is believed that entities having more stakes in the system (e.g., more money involved) are less likely to perform attacks. The mechanism could be seen as a direct democracy in which the entity having the greatest amount of stake involved in the system is the ruler. However, the selection based on the stake of an entity might not always be fair as there may be a dominant entity (e.g. the richest user) which will always make decisions.

The Proof of Stake is faster and less energy-consuming than the Proof of Work but it is at the cost of being less tolerant to attacks.

Delegated Proof of Stake

The **Delegated Proof of Stake** is a mechanism inspired by the Proof of Stake but which differs in the sense that it will be based on a representative democracy [111]. Instead of choosing an entity that will decide, the stakeholders will elect delegates that will generate and validate blocks. The set of nodes that are eligible for signing a block is changed periodically using certain rules. With significantly fewer nodes to validate the block, the block could be confirmed faster. It also leads to a more fair spread of the decision power.

Comparison

The consensus algorithms mentioned above have different advantages and disadvantages [115] in terms of:

- **Tolerated Decisional Power of Adversary**: There are several degrees at which mechanisms are fault-tolerant with respect to the amount of decisional power the adversaries may have.
- **Energy Consumption**: The mining process might be very energy consuming. This may cause several issues regarding the motivation of miners to take part to the consensus process as the cost of electricity might overcome the incentive that they receive. It may also be seen as an environmental issue.
- **Block Generation throughput**: The mining process might cause a bottleneck in the block generation process. This might be an issue for applications needing fast-append operations.

As it is mentioned in *Table 2.1*, Proof of Work protocol has a very high energy footprint [74] as well as not being very energy efficient. Indeed, miners not managing to solve the puzzle in time will lose all the work they have done until then. This may be mitigated by using the work being done for some side-applications. For example, *Primecoin* [66] uses searches for special prime number chains as a puzzle which may then be re-used for mathematical researches.

Furthermore, the mechanism can tolerate the adversaries to have up to 51% of the network's computational power [113, 115]. Indeed, 51% attack has been acknowledged to be a limitation of the

	PoW	PoS	DPoS
Tolerated Power of Adversary	<51% of computing power	<51% of stake	<51% of delegates
Block Throughput	Limited	High	High
Energy Consumption	High	Low	Low
Example	*Bitcoin*, *Ehereum*	*Peercoin*	*Bitshares*

TABLE 2.1: Comparison of the Consensus Mechanisms

Bitcoin technology in the original system [81]. However, it has been proven that even 25% of the network's power would be enough for corrupting the network in some ways [44].

In practice, the use of Proof of Work will result in a limited throughput for block generation. Indeed, the fork phenomenon will be prevented by more difficult puzzles. This causes the search for a solution to be longer and thus, to decrease the chance of fork. However, this limited throughput may be a very restrictive bottleneck for usages needing fast-append to the ledger. [43] is proposing a new design for blockchain needing to face that kind of issues.

On the other hand, Proof of Stake and Delegated Proof of Stake protocols seem to be good energy-saving alternatives as they do not involve any computation. Unfortunately, this advantage comes at the cost of being less resistant to attacks as the mining cost is nearly zero. Indeed, naive proof of stake algorithms are known to be vulnerable to several types of attack/problems such as *Nothing at Stake*, *Long Range Attack* [19]. However, some implementations such as *Blackcoin* [109] and *NovaCoin*[12] use additional mechanisms to address those security issues. Indeed, they use a hybrid consensus mechanism inspired of both Proof of Work and Proof of stake.

Forks

In practice, the entities of the system may decide differently on the block to add to their local blockchain replica. This phenomenon is known as a *fork*. It can be *accidental* or *intentional*:

- *Accidental forks* happen when two entities broadcast a valid block to the network (e.g., In a Proof of Work system, two miners solve the puzzle simultaneously or the broadcast process is slow enough to let another miner generate and broadcast its block).
- *Intentional forks* are the consequences of a change in the consensus rules. Two kinds of intentional forks can be distinguished:
  - *Hard forks* which results from a rule change that causes a software validating blocks according to the old rules to see the blocks produced according to the new rules as invalid.
  - *Soft forks* which results from a rule change that is *backwards-compatible*. Indeed, the blocks generated according to the new rules are recognised as valid by the old software. However, the nodes following the new rules may not see as valid the blocks produced by non-upgraded nodes.

2.3.5 Communication Medium

As mentioned in *Section 2.1.4*, the *CAP Theorem* proved that not all three properties of *consistency*, *availability* and *Partition Tolerance* can be ensured simultaneously by a system. Similarly to most distributed systems, Blockchains are not safe from network failures and thus, should provide guarantees with respect to partition tolerance.

Then, blockchains achieve *eventual consistency* through the use of a broadcast mechanism that will allow nodes of the system to submit data to the network for timely inclusion. From this moment on, the network should converge as quickly as possible to a single valid view of the system. Indeed, if a piece of data is not spread throughout the network quickly enough, the system may reach an inconsistent state in which two sub-networks have a different vision of the blockchain. Moreover, peers should have equal *spreading power*. Actually, if a peer manages to get its messages broadcast more rapidly than others, this could help that peer gain disproportionate profits from deviating from the protocol [44].

2.3.6 Applications & Use Cases

While trends are mainly focused on cryptocurrencies, the use of the blockchain technology is increasing in various fields. In [68], applications in several fields such as *E-government services*, *Health-care*, *Energy* are presented.

E-government services

Blockchain can typically be used to provide e-government services to citizens and businesses. Indeed, it can handle information transactions involving decentralised exchange of digital assets. For example, [87, 101] proposed solutions for making votes transparent and secure in a way that governments are not able to manipulate an election in any kind using blockchain-based electronic voting systems.

Health-care

Health care is another sector that could make use of the blockchain technology. Indeed, the authors of [40, 72] provide a solution to manage *Electronic Medical Record* in a secure, private and simple way using blockchains. More specifically, [40] is presenting prototypes such as *MedRec* and *ARIA* that could be applications of blockchain in Health-care.

Energy

Finally, a field in which blockchain is an emerging technology is the energy market. Indeed, [13, 25, 80, 94] are offering blockchain-based solutions to conduct transparent transactions in the energy market between consumers and the so-called prosumers -i.e., users that both produce and consume energy.

2.4 Background on Bitcoin

2.4.1 General Overview

Bitcoin (BTC) is a purely peer-to-peer (P2P) version of electronic cash [81]. It relies on the blockchain technology to record transactions in a decentralised and distributed way. It allows its users to process payments without the need for any central authority such as financial institutions to coordinate the process.

Electronic Coins Definition

Similarly to [28], the protocol defines a bitcoin as a chain of digital signatures [81]. Each bitcoin has first been *created* and given to a first owner as an incentive of the mining process.
Then, the different owners will transfer fragment [7] or totality of bitcoins to the next by digitally signing: *(1)* a hash of the transaction leading to the owning of the considered coins and, *(2)* the public key of the next owner. Then, the transaction will consist of appending this signature to the end of the coin.

[7] The smaller existing fragment of bitcoin is a **Satoshi** and refers to 0,00000001 bitcoin.

Therefore, a payee can verify the signatures to verify the chain of ownership as illustrated in *Figure 2.10*.

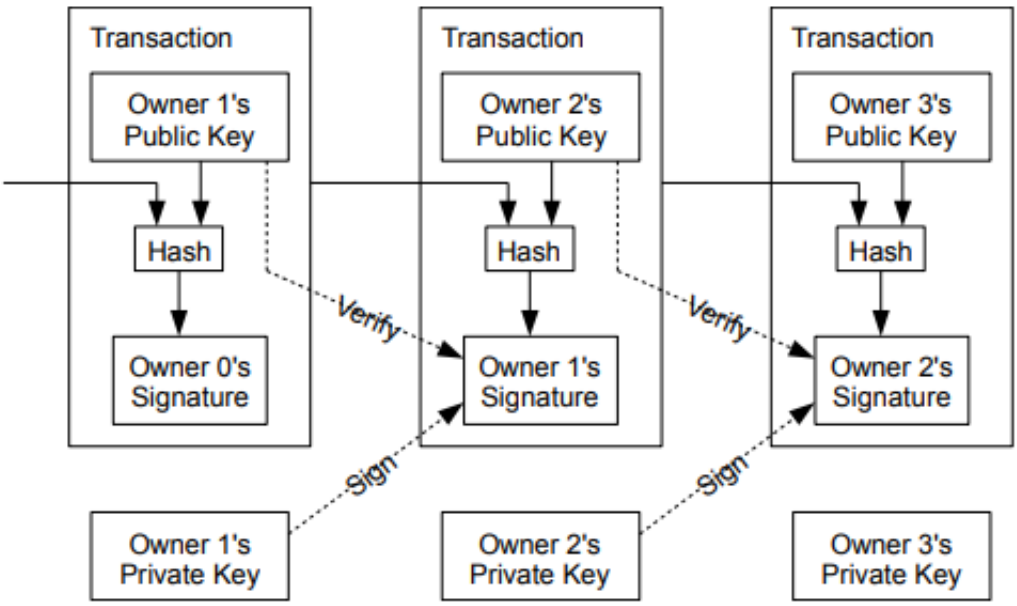


FIGURE 2.10: Bitcoin Transactions [81]

Bitcoin Accounts In practice, users of the system will be identified using one or several *Bitcoin account(s)* which can be defined as an asymmetric cryptography key-pair [106].
A bitcoin account is publicly identified by its public key and users are allowed to send bitcoin to the corresponding bitcoin account using this public information. Then, the private key is needed to spend the bitcoins of an account.

This mechanism provides anonymity to its users as there is no mechanism to link the public key to the physical person though it can be used to identify the owner of coins. Indeed, the public keys can be seen as a pseudonym that can be linked to financial transactions through the public ledger but not to the human identity. [63, 75, 79] are concerned with identifying the owners of those public keys and thus learning the transaction history of users. The most common method used in those works is to analyse transaction patterns (e.g., using clustering techniques) in the public blockchain and link those patterns using other information [63, 75, 79].

Double-Spending Problem The situation in which two or more transactions simultaneously claim the same output is called **double-spending**. Double-spending may be intentional or done by mistake and has been one of the core problems of digital currencies implementation.

Digital signatures only provide part of the solution to the *"double-spending"* problem as they identify the owner of a coin. However, a payee still has no guarantee that a payer has not issued two transactions involving the same coins simultaneously. The real world equivalent of double spending would be a client trying to spend twice the same coins/banknotes.

The starting point of the solution mentioned by Satoshi Nakamoto [81] is that the only way to confirm the absence of a transaction is to be aware of all transactions. This principle is not new and it inspired the most common solution which was to rely on a trusted third-party (e.g., a bank) [28, 93] to keep records of all transactions that have been executed -i.e., a ledger - and to check every transaction

for double spending by analysing this ledger.

The creator of the bitcoin technology chose a totally different approach for solving this challenge. It will rely on an implementation of the distributed ledger technology -i.e., the **blockchain** - to ensure that every node in the system is aware of all transactions that are confirmed. Indeed, the blockchain will be used as a ledger replicated among all nodes and that will track the balance of all accounts in the system. Thus, every peer will be able to check for double-spending before adding a transaction to its ledger without the need for any third-party. However, the participants still need to maintain the replicas of the ledger in a consistent state at all time. To do so, they will need a mechanism to decide on a single common version of the ledger - i.e. a **consensus mechanism**.

Transactions A *Bitcoin transaction* is a movement of bitcoins from a source account to a destination account. An important thing to mention is that transactions are also chained together. Bitcoin wallets softwares give the impression that satoshis are sent from and to wallets but bitcoins actually move from transactions to transactions. A transaction always spends the bitcoins previously received in one or more earlier transactions. Therefore, each input must unambiguously indicate the previous output that leads to the reception of the asset being spent. The only exception being the bitcoins received as incentive of the mining process.

In practice, outputs are tied to **Transaction Identifiers (TXIDs)** which are the hashes of signed transactions. At any given moment, an output may be categorised as either *(1)* **Unspent Transaction Ouputs (UTXOs)** or, *(2)* spent transaction outputs. This ensures that each output of a particular transaction can only be spent once. Thus, for a payment to be valid, it must only use UTXOs as inputs. In addition to that, a payer must perform a digital signature on the transaction using his private key to authorise a bitcoin transfer and proving that he is the real owner of such accounts. This is done with the help of a stack-based scripting language as explained in [82].

Transaction Validation Once a transaction is issued, it will be broadcast to the network for the miners to include it in the block they are mining. However, before doing so, miners will have to validate the transaction and check for fraud. This will be done by *(1)* validating the digital signature with the payer's public key and *(2)* validating that the bitcoins of the input address have not been spent already - i.e., check for *double-spending*. Checking for double spending is done through the analysis of the public ledger. Indeed, the user has a record of all transactions that have been confirmed and can easily track if a coin has not been spent already.

Once a block containing a specific transaction is included in the blockchain, the transaction is said to be a *confirmed transaction*. Still, forks (cfr. *Section 2.3.4*) may happen among the network and may cause the blockchain containing the transaction to be discarded. This is the reason why [**?**] is advising to consider a transaction as confirmed only if the block containing it is 6 blocks deep in the blockchain.

Wallets Practice showed that users are using several accounts -i.e., private-public key pairs to perform their transactions in order to mitigate tracking with user behaviour. Indeed, the re-use of an account for several transactions enables comparison-based attacks on signatures [21] and tracking of coin flows [50, 57, 91]. Therefore, a new key and address should be used for each transaction. *Wallets* have been designed to answer the need for centralised creation and management of those private and public keys.

It is important to mention that the loss of accounts information (such as private keys) leads to the loss of all the coins owned in the name of this account. Indeed, no proof of ownership can be provided to the network and thus, no one will be able to spend those coins.

2.4.2 Blockchain

As mentioned previously, the BTC blockchain permits its users to record an ordered set of transactions in a decentralised and distributed public digital ledger. However, *Section 2.3* specified that blockchain implementations must rely on some components. The three components on which the BTC blockchain is relying are the following:

1. A **Cryptographic Hash Function**: Bitcoin uses **two rounds of SHA-256** as a hash function.
2. A **Consensus Mechanism**: Bitcoin uses an implementation of the **Proof of Work** mechanism: **HashCash**.
3. A **Communication Medium**: Bitcoin blockchain is built on the top of a decentralised **P2P network** that is used to propagate relevant information such as transactions, blockchain updates and consensus information among the peers.

Altogether, those assumptions allow nodes to verify and audit transactions inextensively and to get rid of any central authority coordinating interactions between nodes.

Consensus Mechanism

The BTC blockchain is using an implementation of the Proof of Work mechanism: **HashCash** [14] to ensure the consensus of the nodes on the transactions to append to the ledger.

Hashcash is a Proof-of-Work system that was first used to limit email spam and DDoS attacks but which has then be used as a consensus mechanism for distributed systems such as cryptocurrencies.

The users involved in the consensus mechanism - i.e. the so-called **miners** will compete in finding a solution to HashCash's cryptographic puzzle by following this process:

1. Miners collect unconfirmed transactions that are publicly announced by some nodes of the system.
2. Once it has enough transactions, it will gather all those transactions into a block and will append other information (e.g., a timestamp).
3. Then, the mining procedure will consist of discovering a number - i.e. the so-called **nonce** - such that when included in the appropriate block field, the hash of the entire block has enough zero bits to meet the network difficulty target.

The cryptographic puzzle has an adjustable difficulty - i.e., the number of zeros needed - to compensate increasing hardware speed and varying interest by running nodes over time and thus, ensuring a more or less stable number of block per hour.

The difficulty expected by the bitcoin consensus protocol is updated every 2,016 blocks. The network will use timestamps stored in each block header to calculate the number of seconds elapsed between generations of the first and the last of those 2,016 blocks. The targeted ideal value for 2,016 blocks to be mined is 1.209.600 seconds (i.e., two weeks), it corresponds to a block mined every ten minutes. If the number of seconds is below this threshold, the expected difficulty will be increased proportionally and vice-versa.

Incentives By convention, the first transaction in a block is a special transaction -i.e., the so-called *coinbase transaction* - creating new coins given to the miner as incentive for mining. This is the way new bitcoins are put into circulation. At the same time, there is by conception a fixed number of maximum bitcoin which can be mined: 21 million bitcoins. Thus the amount of new bitcoins which are mined decreases over time, simulating the scarcity of bitcoins and increasing its price, as an analogy to gold mining.

Besides the mining reward, the bitcoin core protocol also defines the usage of transaction fees, which are paid by the transaction issuers to motivate miners to include their transactions in a block they are mining and compensate for decreasing rewards.

Mining Pools A very common practice used for increasing the probability of success in the mining procedure is to regroup and form a coalition - i.e. a **Mining Pool** [30]. Indeed, miners will gather their respective computing power, then agree on a block to mine and finally, cooperate in order to get more chance to win the race and get the incentive. They pool their resources together and split the profit rather than competing for the entire profit.

Mining pools needs a certain degree of exposure but must not appear as having grown too large. Indeed, being able to advertise high win rates can be a useful tool for recruiting other member while approaching a majority of the network's mining power could possibly prevent the rest of the users from globally converging on a growing transaction log [52, 77, 81]. Therefore, it is of critical importance to develop the ability to investigate the true extent of mining pools' collusion.

Block Structure

As in classical blockchain implementations, a block consists of a *block header* and a *block body* that are structured as follows:

- The block header includes:
  - **Block Version**: Indicates the protocol to follow during block validation,
  - **Merkle Tree's Root Hash**: Hash value representing the transactions included in the block.
  - **Timestamp**: The time at which the block was issued.
  - **Difficulty**: The difficulty target computed for this block and used for computing the proof of work.
  - **Nonce**: The solution of the cryptographic puzzle. Used for checking the Proof of Work of the block.
  - **Parent Block's Hash**: A 256-bits hash value used to point to the previous block.
- The block body includes:
  - **Transaction Counter**: The number of transactions that a block contains.
  - **Transaction List**: The transactions included in the block.

It is worth mentioning that the maximum number of transaction that can be included in a block is dependent of the size of each transaction - i.e. number of inputs/outputs.

2.4.3 Network

The Bitcoin blockchain is built on the top of a decentralised, unstructured peer-to-peer network [33, 38, 106], used to propagate relevant information such as transactions between entities, blockchain updates as well as other system information. Indeed, the distributed nature of the blockchain relies on the communication of information between the different users.

Bitcoin Nodes

At first, the bitcoin network was made of very homogeneous peers as the only Bitcoin client available was the reference one. However, the bitcoin network is now made of very heterogeneous peers whose hardware capabilities and software implementations differ largely from each other [33]. In addition to that, different protocols are being used between peers for optimisation of certain tasks.

The nodes can be classified according to the type of services they provide to the network. Indeed, they may be storing different part of the blockchain, may differ considering the protocol they are using when communicating with other peers, considering their connectivity or the functionality they provide to the network. It is important to note that classifying the nodes is a very hard task because of the vast heterogeneity of Bitcoin nodes. Thus, some differences may be found in the real network.

Regarding the storage of the blockchain, we can distinguish three main types of peer: *(1) Full Blockchain* peers (F) that store a complete and up-to-date version of the blockchain[8], *(2) Pruned Blockchain* peers (P) that only store complete blockchain data - i.e. block headers and bodies - for at least the last 2 days[9] and *(3) Header* peers (H) that only store block headers of the full/pruned version of the blockchain.

On the other hand, bitcoin peers may provide the different functionalities such: *(1) Mining* functionalities (M) which consist of creating blocks and working for growing the distributed ledger through the computation of a proof of work and/or *(2) Validation and relay* functionalities (V&R) which consist of validating and relaying transactions, blocks and other network data and/or *(3) Wallets* (W) functionalities which consist of storing sets of keys pairs and track the amount of bitcoins that are associated to those addresses and/or *(4) DNS services* functionalities which consist of informing peers of other existing peers to allow them to connect to those.

Finally, peers running on the bitcoin network may be classified according to their connectivity: *(1) Listening* peers (L) are nodes that accept incoming connections while *(2) Non-Listening* peers (NL) are not (e.g., peers behind NAT and/or firewalls).

An important thing to mention is that even if the original paper [81] implicitly assumed that peers would use a single protocol, the Bitcoin network has grown so big that a lot of different protocols are used by nodes. Indeed, nodes may be following protocols that are optimised for pooled mining (e.g. Getblocktemplate protocol[6] or that speed up data propagation (e.g. FIBRE[5]).

Joining & Maintaining the Network

A typical problem that has to be encountered when dealing with peer-to-peer network is how to make peers join the peer-to-peer network. [110]. A peer joins the peer-to-peer network by establishing a connection to one of the peers that are already connected to the network. Similarly to Gnutella, the bitcoin protocol is not defining any mechanism that should be used by a peer for getting boostrap IPs. In practice, a peer will get those IPs by either *(1)* querying DNS servers that are maintained by

[8] The full blockchain reached a size of approximately 210 GB as of the beginning of April 2019.
[9] The number of days for which a peer is storing complete blockchain data can be tuned.

Node Type	Blockchain Storage	Functionality	Connectivity	Implementations
Full Client	F/P	V&R, W	L/NL	*Satoshi Client* (original), *Bitcore*, *btcd*
SPV Client	H	W		*breadwallet*, *Electrum*, *Simple Bitcoin*, *BitcoinJ*
Non-SPV light Client	-	W	-	*MyCelium*, *Coinomi*, *Copay*
Solo Miner	F/P	V&R, W, M	L/NL	*cgminer*, *BFGMiner*
Pool Mining Server	F/P	V&R, W, M	L/NL	*MPOS*, *CK Pool*
Pool Mining Client	-	W, M	-	*MPOS*, *CK Pool*

TABLE 2.2: Non-exhaustive list of the different types of nodes that are evolving on the bitcoin network and their respective properties - Inspired from [33]

volunteers, *(2)* querying trusted third-parties such as *Bitnodes* [33].

By default, all peers attempt to maintain up to 125 connections and a minimum of 8 connections with other peers: 8 of those connections will be issued by the nodes - i.e, *outbound connections* and up to 117 will be accepted from other peers - i.e., *inbound connections*. The connections are using TCP as a transport layer protocol. However, it is worth mentioning that not all peers allow incoming connection -i.e., connections not initiated by them. Therefore, the Bitcoin network is divided in three subsets:

- **Reachable Network** which refers to listening nodes that use the Bitcoin Protocol or one of its variants. The size of the reachable network is estimated to be varying around 10,000 nodes in 2019 (cfr. **Section 4.2**).
- **Non-Reachable Network** which is made of nodes that are use the Bitcoin Protocol or one of its variants and that may not be listening for/might block incoming connections (e.g. nodes behind NAT/firewalls). [112] aimed at estimating the size of this part of the Bitcoin network and found it to be around 150,000 nodes in 2017.
- **Extended Network** which comprises all nodes of the bitcoin ecosystem that may or not be implementing the Bitcoin Protocol. It includes pooled clients communicating with pool servers using a custom protocol, DNS servers,... No estimation of this part of the Bitcoin network has been found.

Once bootstrap ips have been queried and connections has been established, information on other peers can be queried to *neighbours* -i.e., already connected peers - or can be received spontaneously from them.

Block & Transaction Propagation

The Bitcoin system's primary goal is to provide an *eventually consistent* ordered set of transactions. Therefore, the system must allow its entities to submit transactions for timely inclusion in the blockchain and the system should always converge to a single valid version of this ledger. Bitcoin achieves this consistency through the use of a flooding mechanism which will be responsible for disseminating information among the network as fast as possible. For the purpose of updating and synchronising the

distributed ledger replicas among the system, the two most important information that are disseminated among the bitcoin network are *transactions* and *blocks*..

Transaction is the one most common data structure flowing through the bitcoin network. Indeed, every single node can take part in a transaction by simply using a wallet which is the most basic functionality of a bitcoin node. During May 2019, the bitcoin network was issuing more than 300, 000 transactions per day [3].

On the other hand, blocks are the data structure that contains the transactions issued and confirmed by the network. Unlike transactions, blocks are only generated every 10 minutes which cause them to be flowing less frequently on the network. However, blocks are usually bigger data structure than transactions and outdated nodes may also be asking for specific blocks information for synchronisation. All in all, blocks represent a non-negligible flow in the bitcoin network.

Unfortunately, little of the flooding process is standardised beyond the format of the messages that must be sent. For instance, the protocol is defining a mechanism to avoid sending information to nodes that already received them from other nodes. Indeed, a node can announce the availability of a piece of information to its neighbours -i.e., connected peers - without completely sending it. Then, the remote peer have the possibility to only query the information it didn't received yet. The protocol also planned that the source of a broadcast will not make use of this process when it broadcast brand new information. In this situation, the peer will send the information without prior notice to its neighbour. However, *relays* -i.e., nodes that are not the source of a broadcast - have no obligation to use the mechanism and may sent information to its neighbours directly. In practice, the nodes following the reference implementation blacklist the misbehaving nodes that doesn't follow this protocol.

As of June 2019, the flooding mechanism used by the reference implementation is called the [10]. It spread contents using the previously defined mechanism and by adding independent exponential delays to avoid deanonymisation attacks presented by [17, 69]. This mechanism will be described in more details in following sections.

P2P Network Security & Limitation

The security of the BTC technology is highly dependent on the security of its underlying P2P network. [33, 104] are depicting common attacks that can threaten the bitcoin peer-to-peer network such as *DoS Flooding*, *Eclipse Attack*, *User Profiling*, *Sybil Attack*, *Fake Bootstrapping*. All in all, those threats may cause the whole technology to loose its byzantine fault-tolerance property.

Eclipse Attack This attack was introduced by [58]. In this attack, an attacker is able to control a large number of distinct nodes that populate the whole neighbourhood of the victim node. Therefore, the attacker is able to *eclipse* -i.e., falsified - the view of the network that has the victim. In a cryptocurrency network such as *Bitcoin*, isolating a node from the rest of the network may help the attacker to perform double-spending [58].

Chapter 3

Blockchain P2P Characterisation Methodology

In this chapter, the methodology used for characterising *blockchain peer-to-peer networks* will be presented. Blockchain P2P networks will be characterised through the study of a specific one: the *bitcoin peer-to-peer network*.

The chapter will be structured as follows: In *Section 3.1*, the motivations for the choice of the BTC P2P network as a research subject for characterising blockchain P2P networks will be presented. Then, *Section 3.2* will depict a general overview of the *bitcoin peer-to-peer network*. Finally, *Section 3.3* will explain the methodology used for characterising entities of the bitcoin network as well as a passive method for discovering the *active connections* between those entities.

3.1 Motivations

As mentioned in *Section 2.3*, blockchains can typically be managed by P2P networks. Indeed, they can provide support and substrate to the so-called *Distributed Ledger* by allowing its agents to communicate and exchange information in a decentralised way.

As a matter of fact, the BTC blockchain is one of the most famous and popular distributed ledger using peer-to-peer networks as a support for communication between entities. As a consequence of its popularity, the volume of information flowing through the BTC P2P network, its size and its heterogeneity makes it one of the most relevant blockchain P2P network to be analysed.

Although all blockchain-based technologies make a similar usage of the peer-to-peer network, there is no standard for their design which makes the study of blockchain peer-to-peer network very specific to their implementations. On the other hand, *Bitcoin* has been the first open-source implementation of the blockchain technology and a lot of other blockchain-based technologies such as the *Altcoins* -i.e., variants of the *Bitcoin* crypto-currency are based on its implementation. Indeed, *Altcoins* tried to inspire from the BTC protocol in order to achieve other properties but a deeper analysis showed that network mechanisms are usually kept untouched. For instance, [36] showed that *Litecoin*, *Dogecoin*, *Dash* and *Peercoin* kept exactly the network message types and mechanisms of the BTC protocol.

All in all, those characteristics shows that the reasons why the BTC P2P has been studied so extensively go beyond its popularity and that it is valuable research candidate for characterising blockchain peer-to-peer network.

3.2 General Overview

Bitcoin is using a peer-to-peer network for disseminating different kinds of information such as transactions and blockchain updates among its nodes. The network can be seen as a logical overlay network

which is built up by the BTC software on the top of the IP network and which is dynamically created and managed by its peers. BTC network's formation procedure is intended to induce a random graph topology that should propagate information efficiently. However, a quantitative, thorough measurement and analysis of the BTC P2P network is needed to evaluate if this ideal is actually attained or not.

By default, all nodes of the BTC P2P network are machines running the BTC protocol. It is important to recall that the *nodes* that are highlighted in this studies do not refer to *BTC users* nor *BTC Wallets* as several users, each having several wallets may be connected to the network through a single node. Indeed, most of *BTC users* only use wallets capabilities provided by third-parties which may be connected through 1 or more nodes to the network. The number of *BTC Users* or *BTC Wallets* are better captured through analysis of BTC transactions [57].

3.3 Methodology

The characterisation of the BTC P2P network will be done in two phases :

1. Active Discovery of the nodes that are participating to the network.
2. Passive Topology Discovery of *active connections* between those nodes.

In the first phase of the analysis, a custom *BTC Client* -i.e., the *BTC Crawler* - will be used to crawl the full set of nodes composing the BTC P2P network. Using this crawler, a full snapshot of the entities taking part in the network -i.e., the *active nodes* - can be taken.

Then, the second phase will consist of using the information gathered by the crawler to get a vantage point connected to all the nodes of the network. This will allow the vantage point to passively identify *active links* -i.e., links that has been used during the measure - between *active nodes* and thus, to reconstruct the topology of the BTC P2P network.

3.3.1 Active Node Discovery

The discovery of *active nodes* in blockchain peer-to-peer network has been studied extensively by [46, 83]. Indeed, the characterisation of peer-to-peer network through the analysis of entities is not new and implementations of crawlers already exist for various blockchain-based technologies such as *Ethereum* [65].

The crawler being described in this section has been built from scratch but has been inspired by the experiences of [46, 65, 83]. It is worth mentioning that the technique has been implemented specifically for the BTC P2P Network but are applicable to many other Blockchain networks with similar functioning including *Litecoin*, *Ethereum* and the like.

Problem Definition

The *BTC Crawler* is a customised BTC software client which has both functional and non-functional requirements.

First, the crawler must be able to recursively query all the BTC peers for underlying IP addresses of other peers that has been at some point connected to the network. Then, for research purposes, it should be able to monitor and log measures characterising the BTC entities. For reasons that will be described later, the only measures that will be taken by the crawler as of now are those not needing a significant amount of time to be gathered. However, the crawler may be easily extended to monitor more of the activities of discovered active nodes on a long time basis -e.g., listening to transactions

and blocks that the node is propagating to its neighbours.

The first non-functional requirements that should be mentioned is the crawling speed. Due to the dynamicity of the network, the accuracy of the snapshot taken by the crawler is highly dependant on the time taken to generate it.
For the snapshot to be relevant and as unbiased as possible, the discovery of reachable IP addresses must be as fast as possible. This is especially important as the number of reachable peers is very small when compared to the total number of IP addresses obtained through the peer discovery mechanism.

Then, the crawler when run on simple computers must be able to efficiently store large amount of data on limited-capacity hardware. For instance, it must be able to handle and store several hundred thousands of IP addresses. More precisely, a fast-append and fast-pop structure is needed. Furthermore, the work load sharing among threads and synchronisation must be implemented efficiently to avoid any loss of resources.

As the crawler is participating in the network without directly providing any services[1], resources like bandwidth and processing power required by other peers to serve will be reduced as much as possible. Indeed, the approach, could be considered as a *DoS attack* on the network if scale correctly. Apart from moral reasons, the BTC network has a mechanism to penalise misbehaving node [8] and thus, the crawler should as much as possible stay compliant with the normal behaving of classic BTC nodes and appear as a regular client to other peers to avoid any blacklisting.

Bitcoin Protocol

In this section, the main message types and mechanisms used in the implementation of the crawler will be described. [2, 9] are the main documentations that has been used to describe the protocol. However, it is worth mentioning that the BTC network protocol is not totally specified nor standardised. Therefore, the implementation of the crawler results from the analysis of a combination of documentation, papers, code and tests.

Bootstrap A peer wanting to connect to the BTC P2P network for the first time doesn't know any IP address of peers currently connected to the network. Similarly to *Gnutella* network, the BTC protocol is not specifying the way a peer should be getting seed IP addresses to get connected to the BTC network. In practice, the bootstrapping procedure is done either by:

- Querying a Hardcoded DNS Server -i.e., called *DNS Seeds* - that are maintained by BTC community members. There are two types of DNS Seeds that can be queried : *(1)* Dynamic DNS seed servers which automatically get IP addresses of active nodes by scanning the network and, *(2)* Static DNS seeds servers that are updated manually and are more likely to provide IP addresses for inactive nodes.
- Querying third-parties Servers (e.g., *Bitnodes*) for known active peers' IP addresses.

In either case, DNS seed results are not authenticated and a malicious seed operator or a man-in-the-middle attacker can return IP addresses of nodes controlled by the attacker. This may help a malicious node to perform an attack such as the *Eclipse Attack*. To do so, it will isolate the peer with its own network which will allow the attacker to give the peer a biased vision of the network.

Once the *Bootstrap Phase* has been done, a peer will need to join the network by connecting to one or several of the seed IP addresses it obtained. It is worth mentioning that peers often leave the network or change IP addresses. Therefore, new peers may need several attempts at start-up before

[1] This behaviour is often referred to as *free-riding* [47]

establishing a successful connection. This can add a significant delay to the bootstrapping procedure and may force a user to wait before sending a transaction or checking the status of a payment.

Connecting to peers The BTC P2P network is using the TCP protocol as transport layer protocol to ensure reliable communication between peers. Therefore, for two peer to be connected, they will have to establish a TCP connection as well as going through the BTC application layer's 3-ways handshake.

The BTC 3-ways handshake must initiated by the peer who began the connection. To do so, the peer will have to send a *Version* message to the remote node which contains information about the version of the protocol that it is running, the last block that it received as well as other information such as current time. Then, the remote node will have to acknowledge it received and validate the *Version* message by sending a *Verack* message and its own *Version* message. Finally, both peers will be considered as connected to each other after the peer sent back a *Verack* message.

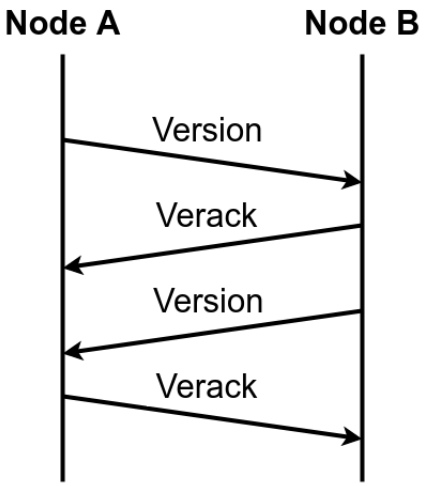


FIGURE 3.1: BTC 3-ways handshake

Peer Discovery Once a peer is connected to another, it can query it for asking the remote node for information about known active peers. This is done by sending a *GetAddr* message to the remote node. Then, the remote node will answer this message by transmitting one or more *Addr* messages each containing up to 1000 tuple *(IP addresses, port)* taken randomly from its database of known peers. The *bitcoin* protocol specifies that each node must keeps a list of up to 2500 peers addresses that it (successfully or not) tried to connect to. This mechanism is providing a fully decentralised method of peer discovery.

Maintaining a connection In order to maintain a connection with a peer, nodes will by default send a *Ping* message to a peers after 30 minutes of inactivity.
If the peer don't react after 90 minutes, the client will assume that the connection has been closed.

Crawler Architecture & Software

As mentioned previously, the crawler is a custom implementation of a *BTC Client* being able to connect and communicate with other *BTC peers* running non-homogeneous protocols.

Functioning In the bootstrap phase, the crawler will obtains a set of *seed* IP addresses by querying all of the following DNS Servers : *(1)* `seed.bitcoin.sipa.be`, *(2)* `dnsseed.bluematt.me`, *(3)* `dnsseed.bitcoin.dashjr.org`, *(4)* `seed.bitcoinstats.com`, *(5)* `seed.bitcoin.jonasschnelli.ch`, *(6)* `seed.`

`btc.petertodd.org`, *(7)* `seed.bitcoin.sprovoost.nl` using the python *dns* package.

Once it is done, the crawling process can starts by first trying to connect to peers among the seed nodes and requesting the reachable remote peer for their internal list of known peers using the *peer discovery* procedure.
Then, the same procedure will be recursively applied for the peer that has been received.

Actually, this approach is not specific to the BTC P2P network and is very common in research concerned with P2P networks or Botnets [99].

Parameters The different parameters that can be provided to the crawler are represented in *Table 3.1*.

Parameters	**Value**	**Comment**
nb_thread	int, $[0, +\infty]$	Number of Threads that will be used by the crawler.
nb_connection_per_thread	int, $[0, +\infty]$	Maximum concurrent number of connections allowed per thread.
time_to_crawl	float, $[0, +\infty]$	Time after which the crawler will be stopped (used for testing purposes).
no_display	True, False	Indicate that no display should be provided during the crawling.
display_progression	True, False	Indicate that the progression of each thread should be displayed during the crawling
network_to_crawl	[ipv4, ipv6, all]	Indicate the Network that will be crawled.
seed_file	string	Name of the file that contain the seed IPs that will be used for crawling.
monitor_connections	True, False	Indicate that all packets exchanged with peers must be stored.

TABLE 3.1: Crawler Parameters

Constants In addition to the parameters that can be provided to the crawler, a set of constants has been chosen and can be configure directly in the source code. The class *Crawler_Constant* is gathering constant that are related to the good performance of the crawler. It contains 4 main constants : *(1) ORIGIN_NETWORK*, *(2) NB_QUERY_PER_PEER*, *(3) CONNECTION_TIMEOUT* and *CONNECTION_ATTEMPTS* while the class *Network_Constant* is mainly used for storing constants related to networking such as socket timeout and message specific timeouts.

The first constant is related to the testing of *BTC Clients*. Indeed, *BTC Core Client* allows to run its client in three types of network : *mainnet*, *testnet* and *regtest*.
The *mainnet* network is the original and main network for BTC transactions, where satoshis have real economic value while the two others are mainly used for testing purposes and satoshis have no real value. Therefore, *ORIGIN_NETWORK* is used to specify which network to crawl.

Then, *NB_QUERY_PER_PEER* is a constant used for calibrating the number of attempts to use the *peer discovery* mechanisms with a peer -i.e., the number of attempts to send a *GetAddr* message. Finally, *CONNECTION_TIMEOUT* is used for limiting the amount of time the crawler will stay connected with a peer and *CONNECTION_ATTEMPTS* for limiting the number of attempts to connect with a remote peer.

IP Storage The crawler has to be using a kind of pool for storing the peers that haven't been processed yet. The characteristics that are wanted for this pool are fast-append and fast-pop.
Furthermore, the number of peers that are stored in this pool may be very large -i.e., several hundreds thousands of (IP, port) tuples and thus, the data structure must be able to efficiently store such data.

To do so, an python implementation of a modified patricia tree data structure -i.e., provided by the python *Pytricia* library has been used for obvious performance reasons [96].

Measurements

The main goal of the BTC crawler is to gather IP addresses and classify those as active or inactive nodes. The BTC protocol is recognising three types of IP addresses : IPv4, IPv6 and OnionCat [1].

OnionCat address format is a way to represent an onion address as an IPv6 address with the first 6 bytes of the OnionCat address being fixed and set to *FD87:D87E:EB43* and the other 10 bytes being the base32 decoded onion address after removing the *.onion* part [1].

Besides collecting IP addresses of both active and inactive nodes in the network, the crawler will also collects several measurements on the peers it manages to connect to :

- Version Measurements : Version of the protocol that the remote peer is running.
  It provide several indications such as the amount of nodes that are up-to-dates or those running custom implementations. The last protocol version available on May the 31st, 2019 is *70015*.
- Service Measurements : Services provided by the remote peer. It provides additional indications aiming at characterising a peer.
  It is worth mentioning that this information is given by the peer itself and can be easily falsified.
  Regarding the reference implementation, several services can be provided by a node such as [2]:
  - *NODE_NETWORK* : The peer advertising this type of service is a full node that can be asked for full blocks -i.e., header and data.
  - *NODE_NETWORK_LIMITED* : The peer advertising this type of service is a full node that can be asked at least for the last 288 full blocks -i.e., more or less the history of the last 2 days.
  - *NODE_BLOOM* : The peer advertising this type of service is a full node capable and willing to handle bloom-filtered connections [54].
    The usage of bloom-filters in distributed systems has already been studied in [105].
  - *NODE_WITNESS* : The peer advertising this type of service is a full node that can be asked for blocks and transactions including witness data.
    Witnesses data are data required to check transaction's validity but not required to determine transaction effects.
  - *NODE_GETUTXO* : The peer advertising this type of service is a full node that can answer to *GETUTXO* protocol request.
    Those nodes are able to answer for request about unspent transaction outputs. It may be useful for lightweight nodes or SPV nodes that doesn't have the full set of unspent transaction outputs at hand and that want to check a transaction for double-spending.

  - *Unspecified* : The peer advertising this type of service is not giving any information on the It may not be able to provide any data except for the transactions it originates.

- Connection Failure Statistics : The reasons why a remote peer disconnected or refused to connect with the crawler.

- Latency Measurements : The monitor peer will record measurements on latencies to remote peers in two distinct ways :

  - by recording the TCP Handshake Duration
  - by recording the BTC Handshake Duration

  Indeed, those mechanisms allows to record measurements on delays without having to introduce any kind of messages in the network nor reducing the crawling speed. Therefore, they are compliant with the idea of parsimonious usage of network resources when not providing any services mentioned in *Section 3.3.1*.

Limitations

Both the nature of the P2P network and the crawling methodology introduce some limitations on the measurements being gathered.

- The number of nodes discovered does not represent the entire network. Indeed, some nodes do not respond to *GetAddr* messages. Therefore, those nodes cannot be queried for information on their neighbourhood. On the other hand, even the standard implementation of the BTC client does not return all the node's neighbours in response to a *GetAddr* queries. Indeed, a peer just answer with a random subset of its database which may limit the information obtained when exploring the network through this mechanism.

- Although there is a reference implementation, the protocol that needs to be followed by a peer is not standardised up to a certain point. Indeed, even in the set of nodes that are up-to-date, there are some differences on the application of the protocol. An illustration of this is the mechanism used by nodes for closing a connection. The reference implementation is mentioned that a connection should be closed by sending back a *Version* message containing the nonce received during the BTC handshake. In practice, not all nodes are using this mechanism and several are sending back *Reject* messages in answer for a request to close the connection.

- BTC nodes may be running on machines using dynamic IP addresses which may cause them to appear once or twice in the set of active peers.

- BTC nodes may be running on machines behind firewalls or NAT which may cause them not to answer the crawler's connections requests.

- The snapshot is not instantaneous. Therefore, the network may change while during the crawling -e.g., nodes considered as active may not still be.

3.3.2 Passive Topology Discovery

The discovery of blockchain peer-to-peer network topology has been the core topic of several studies [35, 78] mainly focusing on the BTC network. However, the distributed ledger technology is in continuous evolution for facing the potential security issues linked to unveiled P2P topologies. This results in constant updates of the underlying protocol, turning some of the previous proposals such as [78] no longer applicable. Indeed, the knowledge of an estimated topology of a given blockchain peer-to-peer network can both be seen as a security issues as well as a mean to evaluate and improve performance of the technology.

Regarding the first usage, one could use the topology to identify and target vulnerable points in a given blockchain network such as peer having low order connectivity to the network. Then, the attacker could use those information for making targeted action such as *Eclipse Attacks* easier. For example, consider two sub-networks of the BTC network that are linked only by one or a few connections. An attacker could easily take control of those few connections and control the blockchain updates that are being transmitted between both networks. Thus, it would be able to double spend transactions on each of the network.

Nevertheless, it may be very useful to acquire this knowledge to efficiently manage the network and optimise its performances. Indeed, the topology could help spot vulnerabilities in the network and thus, to address them. For example, it may be useful for identifying influential node having a high parts of the network' hash power and identify potential entities having more than 50% of the mining power.

In this section, the topology discovery problem in the BTC P2P network will be described and a passive topology discovery method will be introduced. The method takes advantage of the broadcast protocol used to propagate information among the nodes of the network. Applying those results to the real BTC network as well as on other blockchain networks are part of ongoing works.

Bitcoin Broadcast Protocol

As mentioned in *Section 2.3.5*, blockchain networks heavily rely on a broadcast mechanism to disseminate information among the peers. Bitcoin is not an exception to that rule. Indeed, it relies on a flooding mechanism among its underlying P2P network to broadcast data such as blocks and transactions.

In 2015, the Bitcoin community changed the network's flooding mechanism of the reference implementation from a gossip-style protocol known as *Trickle Spreading* to a *Diffusion Spreading* protocol spreading contents with independent exponential delays [10]. Indeed, the practical anonymity implications of transactions broadcasting in the *Trickle* mechanism was challenged by Fanti and Viswanath proposals [17, 69]. However, [17] showed that trickle and diffusion protocols have similar probabilities of detection, both in an asymptotic-order sense and numerically.

The *Diffusion Spreading* mechanism is working as follow: Each block or transaction is introduced to the network at one of the nodes -i.e., its origin. Each origin or relay -i.e., node that is not the source transmitting an information - will transmits the message to each of its *uninfected* neighbours with an independent, exponential delay[2].

In order to avoid sending information to nodes that already received them from other nodes, the relay will previously have to check which neighbours have not been *infected* -i.e., didn't received the piece of information - yet. This will be done by announcing the piece of information through an *Inv* message. Then, neighbours that has not been *infected* will query the corresponding piece of information through a *GetData* message which will be answered by a message containing the piece of information. In practice, several pieces of information may be announced simultaneously within a same *Inv* message. Thus, the *GetData* message is specifying which piece of data has not already been received.

As represented in *Figure 3.2*, the message will be subject to a *Infection Delay* at each hop in the broadcast which is the combination of the *transmission time* over the peer-to-peer network and the *local verification of the block/transaction*. The *transmission time* includes the exponential delay induced by the *Diffusion Spreading* mechanism, an announcement in the form of an *Inv* message, a request from the

[2]In practice, higher rate are applied on *outgoing* edges -i.e., connection established by the peer - than *incoming* ones - i.e., connection established by the remote peer [11]

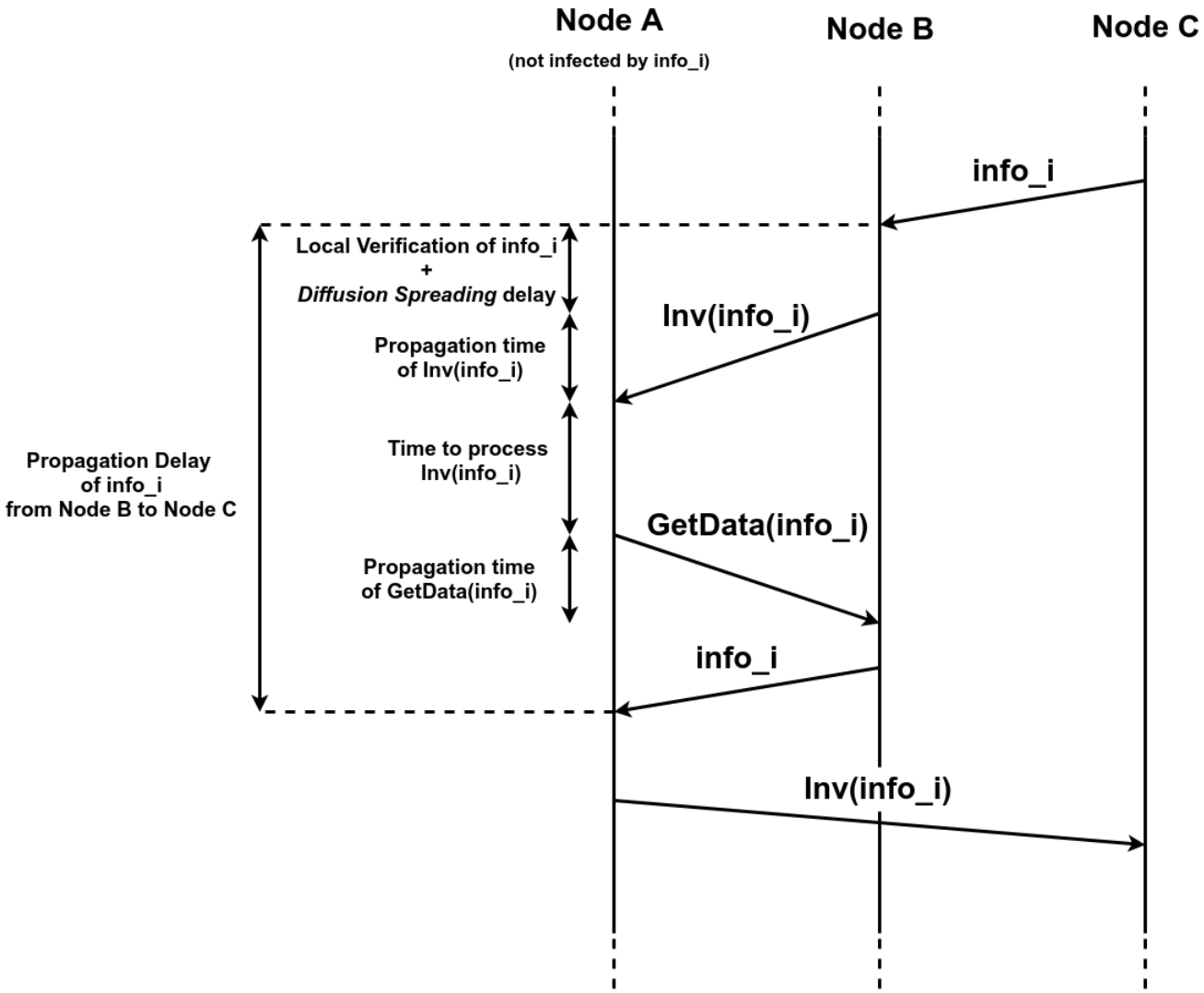


FIGURE 3.2: BTC Broadcast Protocol

receiving party in the form of a *GetData* and the delivery of the requested data.

Problem Definition

As mentioned previously, the formation procedure of the BTC P2P network intend to induce a random graph and the network is dynamically created and managed by its peers without having any central entity organising nor coordinating the network. Therefore, the BTC network will be assimilated to a random graph with its set of vertices representing the *active nodes* of the network and the set of edges representing the connections between those nodes.

In practice, a node is only considered as *active* -i.e., part of the BTC network - if there is a path connecting it to all the other active nodes of the network. Therefore, the graph representing the BTC network must be connected.

In addition to that, the connection between two active peers is always bidirectional which could lead the network to be seen as an undirected graph. However, a connection is having propagation

delays differing depending on whether the message is flowing in one direction or the other. Therefore, the BTC P2P network will instead be seen as a directed graph with each edge's weight being the propagation delay of the corresponding directed connection.

All together, the *bitcoin peer-to-peer* network can be modelled as random, connected, undirected, weighted graph and the topology discovery problem will consists in discovering the presence or absence of edges while the set V of active nodes is known.

The problem of discovering the set of *active nodes* has already been studied in the previous section. Therefore, the main problem that will be addressed in this section is the one of finding the presence or absence of edges. The method will consider that all edges are not present by default and then will try to infer which edges exists.

To do so, it will rely on a *supernode* -i.e., a node connected to all the active nodes in the network - to monitor the propagation of information among the network. Then, by re-crossing all the information gathered, it will try to infer the path that the different messages took.

Network Model Formalisation

The BTC network will be modelled by a random, connected, undirected, weighted graph $G(V, E)$ with V being the set of vertices in the graph corresponding to the *active nodes* of the real network and E the set of edges in the graph corresponding to the connections between those nodes. $(u, v) \in E$ is an edge of G joining vertices $u \in V$ and $v \in V$ which represents a connections between the corresponding active nodes in the real network.

The weight of each edge $w_{(u,v),t}$ represents the *propagation time* of a broadcast message relayed by node u to node v at time t in the network. However, the propagation time across a connection is not constant and can be modelled by a stochastic process $\Sigma_{u,v}$ being the sum of 3 stochastic processes :

- $\Delta_{u,v}$ that is a stochastic process representing the propagation delay between two nodes u and v across the network. This is mainly used to represent the jitter -i.e., fluctuation of delays - in the BTC network.
- $\Gamma_{u,v}$ that is a stochastic process representing the overall processing time induced by the relaying of information in the BTC network. It includes the local verification of the piece of information (e.g., blocks/transactions) that is relayed by the sending node, the processing of potential *Inv* messages at the receiving node, the potential creation of a request from the receiving party in the form of a *GetData* message and the creation of the message gathering the data asked.
- $\Lambda_{u,v}$ that is a stochastic process representing the delay induced by the *Diffusion Spreading* mechanism. The reference implementation mentioned that this delay is following an exponential distribution.

For the sake of simplicity, the 3 stochastic processes are considered stationary and independent. Thus, their unconditional joint probability distribution does not change when shifted in time and parameters such as mean and variance also do not change over time. As a consequence of that, the *propagation time* across a connection resulting from the sum of these 3 processes can be modelled by a stationary stochastic process (cfr. *Appendix A.1*). In addition to that, the assumption that $w_{(u,v),t}, (u, v) \in E$ and $w_{(k,l),t}, (k, l) \in E$ are independent if $(u, v) ! = (k, l)$ will be assumed.

Let $P_{u,v}$ denote the set of path between nodes u and v. A path $p \in P_{u,v}$ between u and v is called simple if p does not cross itself. The set of simple path between nodes u and v will be denoted by $S_{u,v}$. Then, $[u, v]_t$ will denote any shortest path between u and v at time t and $d_t(u, v)$ be the length of a shortest path between nodes u and v such that:

$$[u,v]_t = \min_{p \in S_{u,v}} \sum_{(k,l) \in p} w_{(k,l),t} \tag{3.1}$$

$$d_t(u,v) = \sum_{(k,l) \in [u,v]_t} w_{(k,l),t} \tag{3.2}$$

For each vertex $v \in V$, $d_{u,t}(v) = d_t(u,v)$ will denote the distance function from a node $u \in V$ to any other node $v \in V$ at time t. Therefore, $d_{u,t}(v)$ denotes the stochastic process resulting from the sum of the stochastic processes representing each edges along the shortest path $[u,v]_t$.

The broadcast of an information among the network by a single source vertex u can be modelled by a *cascade*. A *cascade* consists of single source vertex u that initiates an information diffusion together with the times $\{T_{u,t}(v) : v \in V\}$, where $T_{u,t}(v)$ is the amount of time it takes information to propagate from vertex u to vertex v at time t. In practice, a BTC node will only consider the first copy of a message that it received and will ignore any duplicate. Therefore, the time $T_{u,t}(v)$ can be associated to the distance function $d_{u,t}(v)$ as the first copy must have taken the shortest path.

Methodology

The methodology considers a *supernode* which is able to monitor the propagation of broadcast messages and record the time at which it received a copy of the information from each of the active nodes in the network.

Then, the method proposed in this section will assume that, for each broadcast message, it is able to infer the source of the message and the time at which it initiated the broadcast process. This process is known as *deanonymisation* and has been the topic of several works such as [18, 45, 69]. In addition to that, the method also assumes that all messages are relayed using the *Diffusion Spreading* mechanism described previously and that all connections can be represented by totally independent stochastic processes.

Considering all that has been said, the topology discovery method will work as follow :

1. For each information that has been broadcast :
   (a) Record the times t_v at which the *supernode* received the information from each of the active nodes v in the network.
   (b) Infer the source of the broadcast u and the time at which it initiated the broadcast t_0.
   (c) Infer the propagation time of the path starting at source node u to the *supernode* s through node v.

$$T_{u,t_v}(v) - d_{v,t_v}(s) = t_v - t_0 \tag{3.3}$$

   (d) Using previously recorded information, try to infer the tree representing the broadcast's propagation in the network.

The source of the broadcast u and the set of propagation time $\{T_{u,t_v}(v) - d_{v,t_v}(s), v \in V\}$ represents a *cascade* among the graph modelling the BTC network. Therefore, the problem to solve is the one of network topology inference using information cascades. This problem has been studied extensively in the graph theory literature by [23, 60, 62].

Network Topology Inference using Information Cascades

Once a set of cascades $\{u, \{T_{u,t_v}(v)\, v \in V, v \neq u\}, u \in U\}$ has been discovered, the method will try to infer the corresponding trees by re-crossing the cascades information.

First, a time series $\{T_{u,t}(v)\}$ will be constructed from the set of cascades for each shortest path $[u,v]_t$ gathering all the information monitored on the path. Each element of these time series will represents the distance measure of the shortest path between two nodes u and v at a time t which is the sum of the realisation of the stochastic processes of the edges belonging to the path at a time t.

However, it has been mentioned that two connections (u,v) and (k,l) belonging to the BTC network are assumed to be independent. Therefore, the network topology inference problem can be broken down to the problem of finding the set of edges having the least amount of dependence between its time series leading to a connected graph and that explains the set of cascades that has been monitored.

To do so, a measure of how much the time series are dependent with each other will be computed.

$$dependence(d_{u,t}(v), d_{k,t}(l)) \approx dependence(\{T_{u,t}(v)\}, \{T_{k,t}(l)\}) \tag{3.4}$$

Indeed, this measure represents an approximation of the dependence between the stochastic processes representing the shortest paths $[u,v]_t$ and $[k,l]_t$.

Then, for each time series $\{T_{u,t}(v)\}$, the mean of the dependence with all other time series will be computed

$$dependence(d_{u,t}(v)) \approx \underset{(k,l)\in E,(u,v)\neq(k,l)}{mean}(dependence(\{T_{u,t}(v)\}, \{T_{k,t}(l)\})) \tag{3.5}$$

Finally, the set of broadcast path will be reconstruct in the following way:

1. For each information broadcast from node $u, u \in V$ among the network,
   (a) The path p from node u to node $v \in V$ is the path such that

$$p = \underset{p\in S_{u,v}}{min}\ \underset{(k,l)\in p}{mean}(dependence(d_{k,t}(l))) \tag{3.6}$$

It is worth mentioning that the method will only attempt in reconstructing the path of the broadcast it monitored. Therefore, only *active edges* -i.e., edges that has been used in one or several broadcast - will be part of the reconstruction.

Dependence Measure

The independence of time series has been studied extensively in the literature [41, 67, 85, 92, 103].

The time series considered in this section are the realisation of a *stationary stochastic processes*. Indeed, the stochastic process representing a path in the BTC network is the sum of the stochastic processes representing each edges along the path which all are stationary. All in all, the dependence measure must search for linear dependencies among a set of stationary time series

Limitation

The model proposed in this section introduce some limitations regarding its practical implication:

- The set of nodes that are source of a cascade is often a very small subset of the entire network. Indeed, [37] showed that among 13000 different nodes, 20 nodes were responsible for first relaying more than 70% of blocks and transactions in the BTC network.

- Staying connected to all nodes for an extensive amount of time is a difficult task. In practice, the set of nodes that are reachable and allows to connect to them is only a subset of the entire BTC network.
- Regarding *Diffusion Spreading*, all nodes may not follow the reference implementation and may be using different kind of distribution for producing their artificial delays.
- The method only aims at discovering *active edges* which has been used during the monitoring.
- There are no obligations for a node to relay a message. Therefore, complete cascade information may not be accessible.
- Stochastic processes representing edges of the BTC network are not totally independent. Indeed, the connections (u, v) and (k, l) of the BTC P2P network may share edges on the underlying IP network. Therefore, the stochastic processes $\Delta_{u,v}$ and $\Delta_{k,l}$ representing the propagation delays across the connections will not be independent.

Chapter 4

Results

In this chapter, the results obtained using the implemented *BTC crawler* will be presented.

The chapter is structured as follows: In *Section 4.1*, the results obtained with the aforementioned *BTC crawler* as well as the methodology for generating those results will be presented. Then, *Section 4.1* will describe a temporal characterisation of the BTC network sing the publicly available data.

4.1 Single Snapshot Analysis

4.1.1 Results Generation Methodology

The two snapshots of the BTC P2P network that are presented in this section have been obtained by running the aforementioned crawler using two different vantage points both in terms of hardware and of geographical location.

Crawler Parameters The first parameters that will be discussed is the interval at which the crawler will be sending *GetAddr* message to discover the neighbourhood of a peer. Indeed, a tradeoff between the number of IP addresses will be queried and the effort (e.g., bandwidth usage, processing time) created at the remote peer must be decided. In accordance with the principle of minimising, the effect of the crawling on other peers and the risks of being blacklisted while keeping a satisfying crawling speed, the value of one *GetAddr* message every twenty seconds with a maximum of two queries per remote peers has been chosen. As IP addresses may also be announced unsolicited by remote peers, the configured request frequency seemed to be sufficient for the crawler.

Similarly, the number of connection attempts should also be limited. For the sake of crawling speed, the results were generated imposing a single attempts of connection per peer when running the crawler with laptop and two attempts when considering the cluster.
However, those values should be discussed in more details as peer may appear as unreachable for a period of time (e.g., they reached their limit in terms of connections) and then be reachable again.

Then, the number of threads that are simultaneously establishing connections to remote peers should also be discussed. The crawler when run on the cluster used 500 threads while the laptop was only using 10 threads.

4.1.2 Snapshot taken on September the 10^{th}, 2018

The first snapshot that will be discussed is one taken by the crawler on September the 10^{th}, 2018 using two topologically co-located vantage points in *Austria* and *Germany* provided by the *Austrian Institute of Technology*. The snapshots from both vantage points has been compared empirically and seems to be equivalent.

The crawler discovered more than 200.000 bitcoin IPs with a total of 9,240 active nodes among which 87% corresponded to IPv4 addresses, 13% to IPv6 addresses. Then, the data has been complemented using the *bitnodes* API [7] for identifying 433 anonymous connections being done using the *Tor* services. All together, 9,673 active nodes has been identified, 83% being IPv4 addresses, 13% being IPv6 addresses and the remaining 4% being the *.onion* addresses. Then, both using the *MaxMind*and *Team Cymru* [114] IP-geo-localisation databases, the IPv4 and IPv6 addresses has been geo-located by country and by ASes hosting them. Finally, the public bitcoin blockchain ledger of transactions will be processed to infer the identity of miners since early 2017 until end 2018. Indeed, when a new block is mined, the miner usually signs the associated *coinbase transaction* -i.e., first transaction of any blocks used for the miners to claim its reward - with its name providing as such a signature for his activity. However, it is worth mentioning that adding such a signature is non compulsory and can be easily falsified.

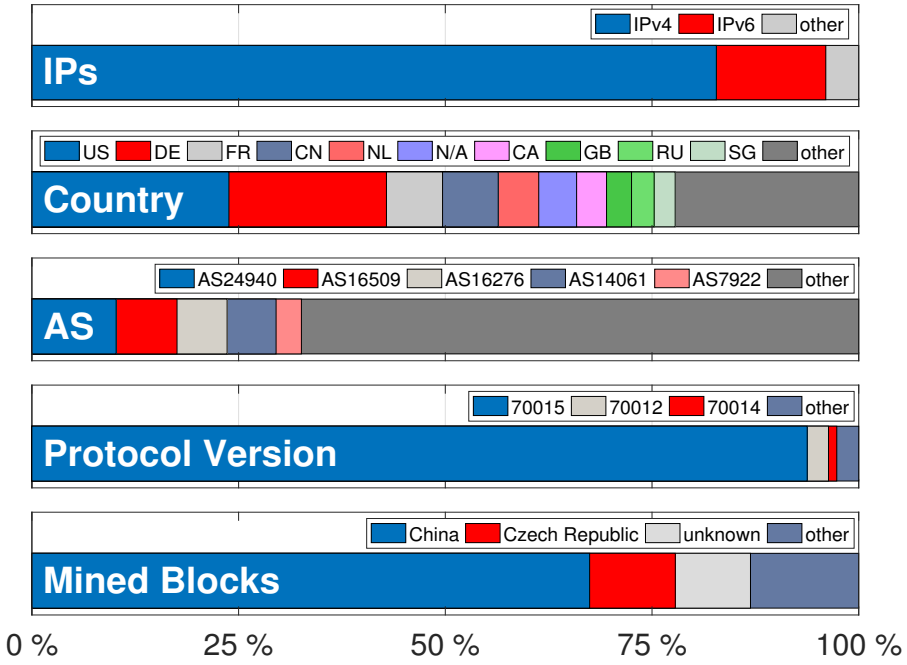


FIGURE 4.1: Results of the crawler run on September the 10th, 2018.

Figure 4.1 presents all the statistics gathered during the crawling. The first thing that can be observed is that most of the BTC active nodes are located in the United-States(23.7%) and Europe -i.e., Germany(19%), France(6,8%), Netherlands(4,9%) whereas a smaller share are located in China (6,7%) and other Asian countries (Singapore, Japan, South Korea, etc.). Moreover, the figure also demonstrates that the sample that has been collected is broad and not limited to a specific part of the Bitcoin network.

This is additionally confirmed by the propagation latency -i.e., the minimum Round Trip Time (RTT) - to the corresponding IPs [86], from one of the vantage points used in the data collection located in Austria.

Indeed, *Figure 4.2* shows that about 50% of the IPs reside within Europe (min RTT < 50 ms), 30% within North-America (100ms < min RTT < 200ms) and about 10% at very far locations such as eastern Asia (e.g., Japan, Korea, Hong Kong, etc.).

Regarding the hosting ASes, a big share of nodes are hosted by major cloud providers such as *Hetzner* (AS24940), *Amazon* (AS16509) and *OVH* (AS16276). Despite such a western concentration of active nodes, it is impressive to see that most of the the BTC mining activity actually occurs in China.

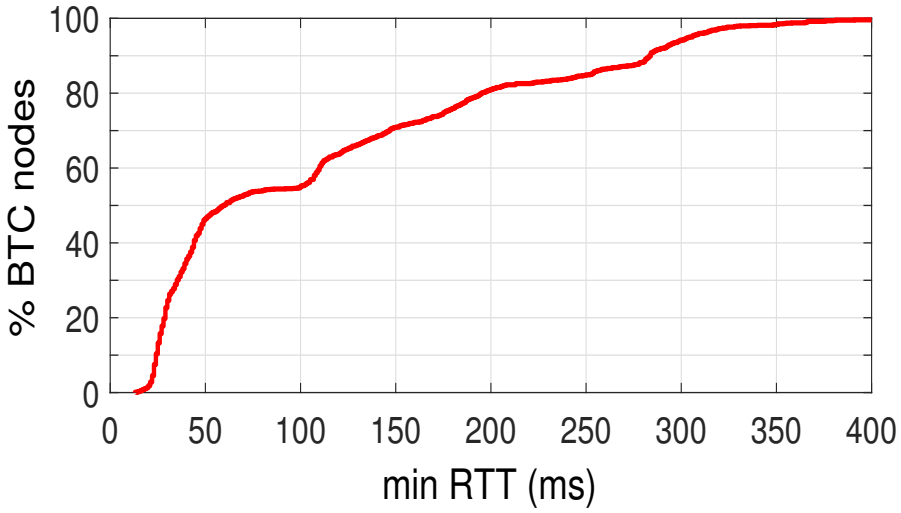


FIGURE 4.2: Minimum Round Trip Time to active BTC nodes

Indeed, using the signature provided by the miners when producing a new block, it can be observed that more than 70% of the BTC mining activity is controlled by major Chinese mining pools. The number of blocks mined by unknown or hidden miners is also non-negligible as it represents about 7% of the mined blocks.

Concerning the nodes services, the crawling measures shows that the biggest majority of the BTC nodes keep a full copy of the full BTC distributed ledger (93,6%) and can therefore perform full validation of transactions and blocks. On the other hand, about 93,8% of nodes use the latest version of the BTC protocol while there are at least 2,6% of the nodes using a very out-dated protocol dating from late 2015. This may potentially result in security or performance issued.

The measurements shows that the BTC P2P infrastructure is mainly hosted in western cloud providers and ISPs with a quite centralised infrastructure deployed at few major providers. This may puts into quetion the decentralisation nature of the BTC network.
In terms of mining activity, the picture looks even more critical as about 70% of BTC blocks are mined by major Chinese pools. Indeed, this raises concerns with the very famous security threats in terms of BTC blockchain controls - the so-called 51% attack [104].

4.1.3 Snapshot taken on May the 31st, 2018

In this section, the results obtained when crawling using a *Acer Nitro 5 AN515-51* laptop with an Intel Core i7 Processor (4x 2.8 GHz) and 4 GB DDR4 RAM located in *Liège*, *Belgium* as a vantage point will be described.

The crawler collected around 200,000 ips among which there was 75.4119% IPv4 addresses, 23.8905% IPv6 addresses and 0.6976% OnionCat addresses. Then, the 6,819 peers that has been considered as active by the crawler will be complemented with the 232 Tor nodes identified by the *bitnodes* third-party which results in 7051 active peers collected, with 81.3785% being IPv4 addresses, 15.3312% being IPv6 addresses and 3.2903% being Tor nodes.

Figure 4.3 presents the statistics gathered during the crawling. First, it can be observed that most of the BTC nodes are still identified as being located in the United-States (24,725%) and in Europe -i.e., Germany (18,62%), France (6%), Netherlands (5,3%) whereas a smaller share are still geo-located in

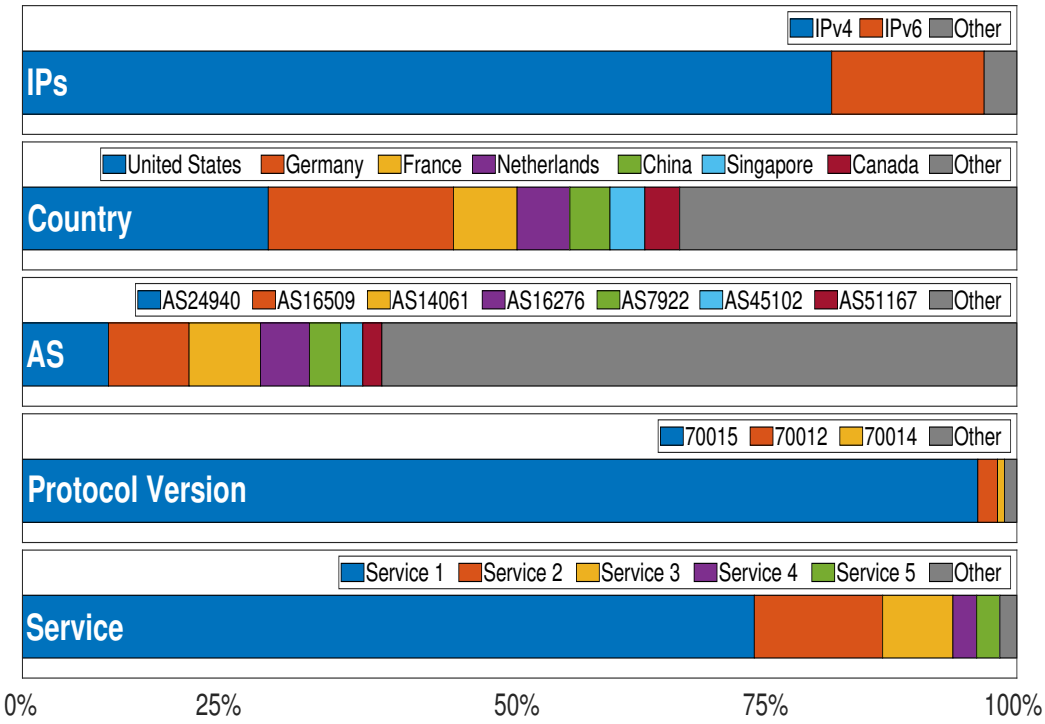


FIGURE 4.3: Results of the crawler run on May the 31st, 2019.

China (4%). Then, the majority of hosting ASes are still *Hetzner* (AS24940), *Amazon* (AS16509) respectively with shares being 8,65% and 8,11%.

Regarding the protocol version statistics, it seems that about 96,07% nodes uses the latest version of the BTC protocol while there are at least 2.7% of nodes using a very out-dated protocol dating from late 2015. On the other hand, it seems that about 88,83% of nodes keep a full copy of the full BTC distributed ledger. Indeed, the services mentioned in the above figure are labelled as follow (cfr. *Section 3.3.1 Measurements*):

1. Service1 = NODE_NETWORK + NODE_NETWORK_LIMITED + NODE_WITNESS + NODE_BLOOM
2. Service2 = NODE_NETWORK + NODE_WITNESS + NODE_BLOOM
3. Service3 = NODE_NETWORK_LIMITED + NODE_WITNESS + NODE_BLOOM
4. Service4 = NODE_WITNESS + NODE_BLOOM
5. Service5 = NODE_NETWORK + NODE_BLOOM

Finally, the measures shows a difference of about 2,000 nodes that are considered as active between the first and the second snapshot. However, *bitnodes* API detected 9,357 active nodes in the time of the second snapshot. This gives indications that the second snapshot may be less accurate than the first one.

The difference in accuracy between both snapshots can be explained by two things: *(1)* the lower number of connections attempts or, *(2)* the difference in terms of geographical location. This could be the topic of further investigations.

On the other hand, the measures seems to be very similar which gives additional confidence in the crawling methodology.

4.2 Longitudinal Analysis

In this section, a temporal characterisation of the BTC network will be given using the publicly available API provided by *bitnodes* [7] for populating the dataset. The *bitnodes* API is providing a similar dataset as the one generated through the crawler, providing a full snapshot every five minutes for the last 60 days. A dataset spanning 110 consecutive days, starting in February 2019 with a full BTC snapshot every 30 minutes has been constructed by querying the third-party. Then, publicly available data on the BTC ledger has been queried from both [3] and [4] APIs.

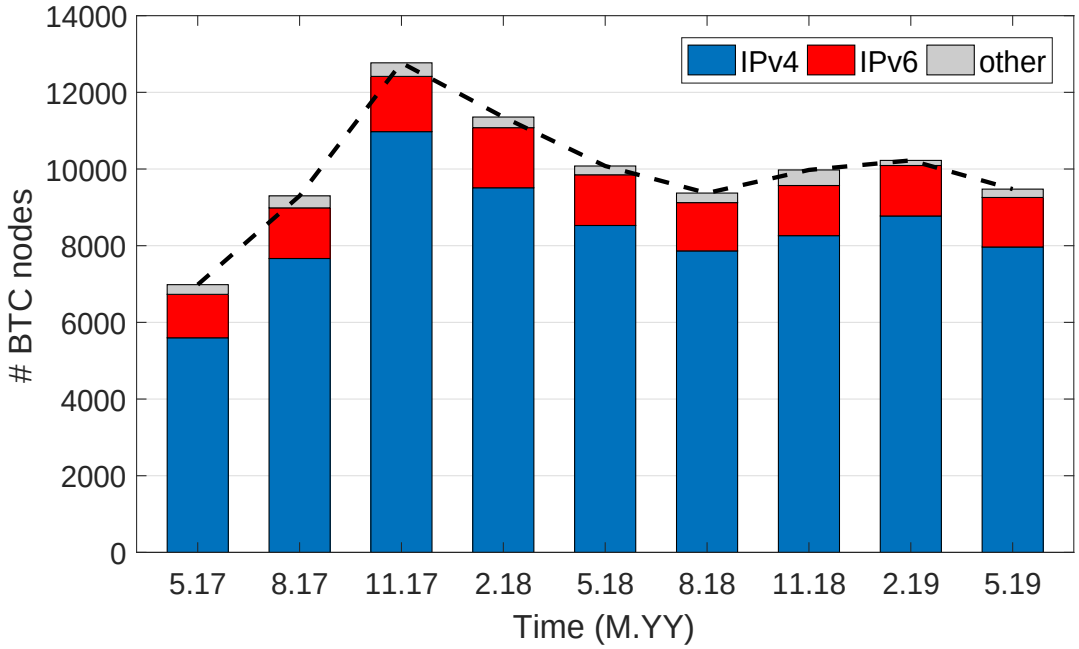


FIGURE 4.4: Number of active BTC nodes along time

While the numbers of active nodes grew significantly in 2017 -i.e., almost doubling in less than 6 months, *Figure 4.4* shows that from the end of the bubble the size of the BTC active network has remained almost constant during the past year, with about 10,000 nodes daily active. This suggests that the number of users interested in running BTC nodes is not growing and that the underlying P2P network is rather stable in terms of new members.

Then, *Figure 4.5* plots the temporal evolution of the share of mined blocks, during the past 2 years, until the end of 2018. The op mining pools by the end of 2018 in terms of blocks is lead by major Chinese companies such as AntPool (12.3%), BTC.com (18.2%), ViaBTC (9.9%), BTC.top (8%), F2Pool (8%), and Poolin (7.6%), keeping a similar dominance along time. Other non-Chinese pools include SlushPool (12.2%) in Czech Republic and Bit-Fury (2.3%) in Georgia.

To sum up, the temporal analysis reveals that the size of the BTC P2P network has remained constant over the past year and that the Chinese dominance in terms of mining activity holds along time, with a very centralized structure. The main four Chinese mining pools have covered more than 50% of the mining activity since roughly mid 2017, which is a serious security threat for the integrity and reliability of the whole BTC network.

Nodes Performance Analysis Finally, the last part of the study is devoted to the analysis of the performance of each BTC nodes. To do so, the BTC Node Index (BNI), a metric inspired from one seen in the *bitnodes* project which considers multiple properties of a node to rank its goodness as part of the

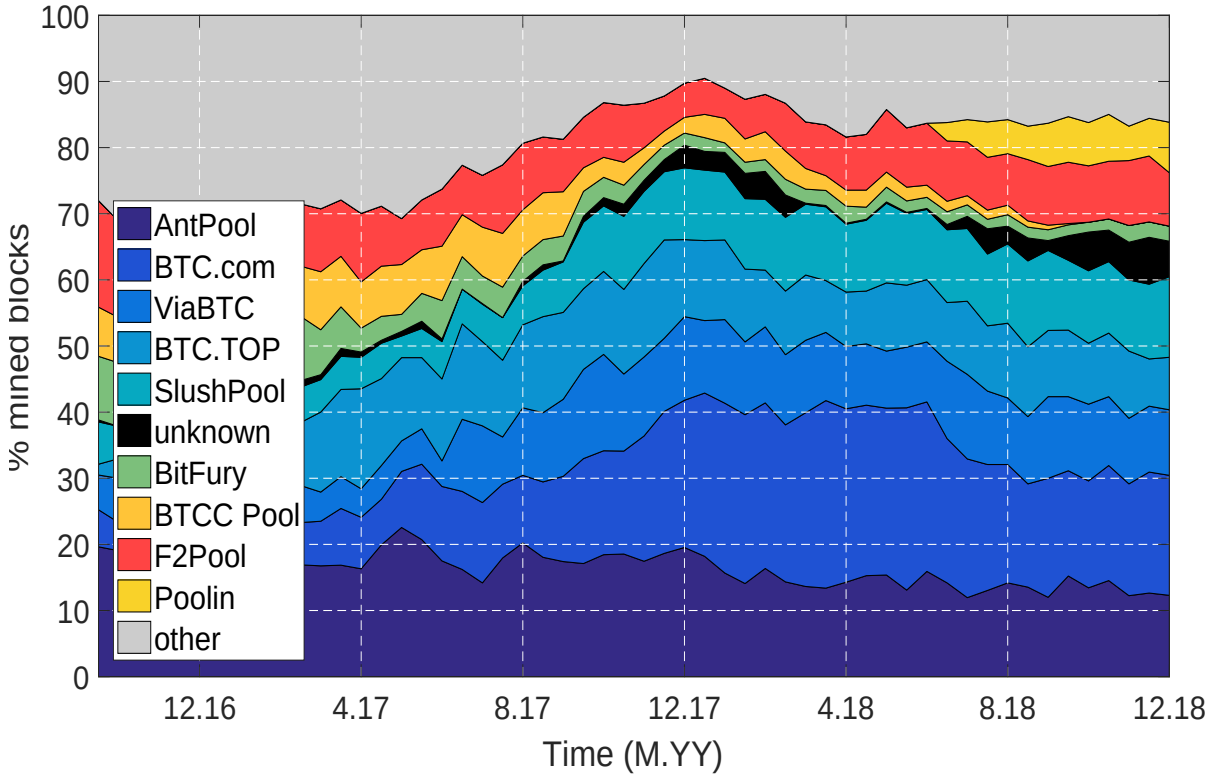


FIGURE 4.5: Evolution of share of mined blocks among pools and single miners

BTC network will be defined.

The BNI index ranges from 0 to 10 with 10 representing the best fitted node for the BTC network and 0 the worst. It is is computed as an average of 10 different node-to-network metrics which basically reflect how similar is a node to the majority or to the most common node as well as how good is this node connected and synchronised to the network:

1. Protocol Version index = 1/r, *'r'* being the rank of the node's protocol version (e.g., it is equal to 1 if the node is running the most common protocol)

2. Service index = 1/r, *'r'* being the rank of the node's service (e.g., It is equal to 1 if the node is providing the most common service).
   Note that a node not providing *NODE_NETWORK* services has automatically its service index equal to 0.

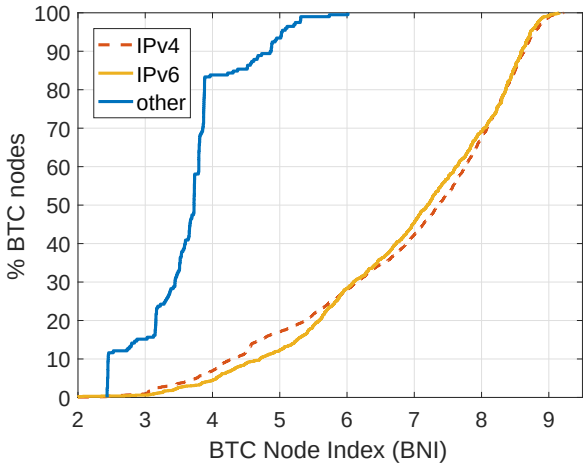
3. ASN index = ln((1 / n) x N) / ln(N), *'N'* being the number of reachable nodes and *'n'* being the number of nodes from N with the same ASN.

4. Port index = 1 if the node is reachable through the default port -i.e., 833 and 0 otherwise.

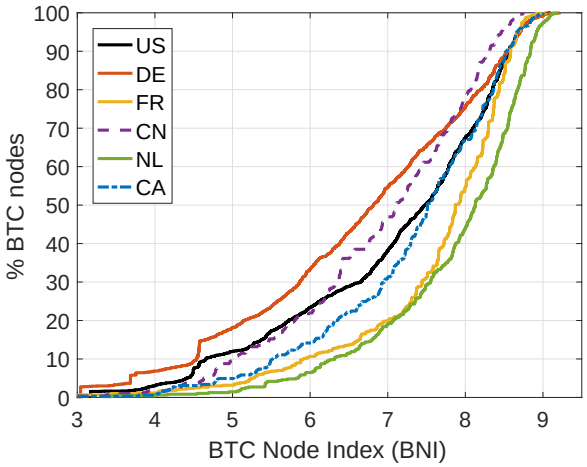
5. Connected Since Index = 1/r, *'r'* being the rank of the node's connection's start timestamp. Connection's start timestamp is ranked 1 followed by the next oldest timestamp. (peer with the oldest connection's start timestamp (highest connection duration) is ranked 1)

All together, those metrics assess the *quality* of a node with respect to the standards.

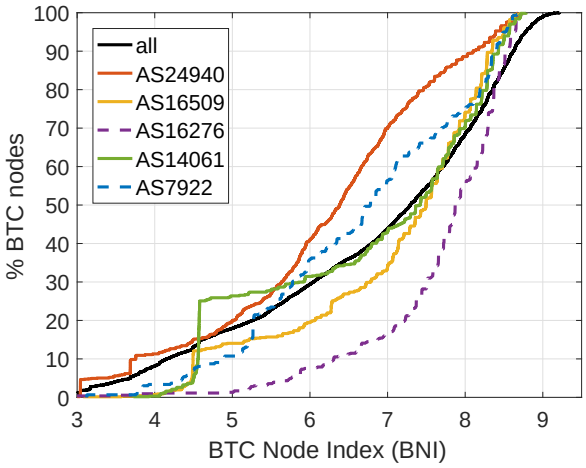
Figure 4.6 reports the BNI of the network for the last week of the dataset, splitting by *(a)* network type, *(b)* top ASes and *(c)* top countries. It shows that there is no significant difference between IPv4 and IPv6 nodes while Tor nodes are significantly worse ranked. This is mainly due to their mismatch



(A) BNI per network type



(B) BNI per top-AS



(C) BNI per top-country

FIGURE 4.6: Bitcoin Node Index (BNI). The BNI index aggregated 10 different node-to-network metrics.

to the majority of nodes in terms of node properties.
In terms of ASes, nodes hosted by OVH and Amazon outperform those hosted by other main ASes with a significant difference between OVH and Hetzner which performs the worst.
Finally nodes hosted at the Netherlands and France systematically rank higher than those hosted in United-States and China. The analysis of nodes' quality can be the topic of future work.

Chapter 5

Conclusion

First, the thesis provided a survey on the blockchain technology and its implementation: the *Bitcoin*. Then, the BTC P2P network was characterised by analysing its nodes from a purely network measurements-based approach. By crawling and locating the active BTC nodes, as well as by studying the associated BTC mining activity, the following conclusions have been drawn: *(1)* Despite the fuss around BTC and crypto-currencies, the size of the active BTC network has remained rather fixed right after the main drop in BTC price, in early 2018, *(2)* the BTC network is mainly located in western countries, being the United-States, Germany and France the dominant hosting countries, with more than 50% of the active nodes and, *(3)* Despite this western network locality, more than 65% of the BTC blocks are mined by major Chinese mining pools, calling for potential centralisation and blockchain immutability/security issues. Finally, a passive topology discovery approach for blockchain P2P networks and its requirements has been introduced.

5.1 Future Works

Several lines of research arise from this work and are worth being pursued.

Firstly, optimisation of the crawler for improved performance at taking snapshots of the BTC P2P network could be investigated. Indeed, the relevance of the BTC snapshots relies on the fast and accurate crawling of the BTC P2P network. Then, a deeper analysis of the BTC entities and their *quality* as described in *Section 4.2* could be done by using a dataset covering a longer period.

Regarding the unveiling of the BTC P2P network topology discovery, a Proof of Concept under controlled simulated environments is part of ongoing work. Indeed, the method is being studied both from a graph theory point of view as well as when used on a real simulated blockchain local network. Then, simulations must be done using different kinds of topologies to ensure that the method is general. Finally, tests on the real BTC network must be done by using ground-truth nodes to check if the method succeeds at discovering the outgoing and incoming connections of those nodes.

5.2 Publications

In the context of this Master's thesis, two papers have been written:

- **Vivisecting Blockchain P2P Networks - Unveiling the Bitcoin IP Network** (CoNEXT 2018)
  S. Ben Mariem, P. Casas, B. Donnet
- **O Bitcoin Where Art Thou? - Unveiling the Bitcoin IP Network** (IMC 2019)
  S. Ben Mariem, P. Casas, M. Romiti, B. Donnet, B. Haslhofer, R. Stütz

One of them is currently under review at the upcoming *ACM Internet Measurement Conference* (IMC 2019) while the other has been published at the *ACM International Conference on emerging Networking EXperiments and Technologies* (CoNEXT 2018).

Appendix A

Appendix

A.1 Proof that the sum of two independent stationary stochastic processes is a stationary process

Let ϵ_t and ν_s two independent stationary process. Show that $\{\epsilon_t + \nu_s\}$ is a stationary process.

Proof:

$$\begin{cases} E[\epsilon_t] = \mu_1 \\ Var(\epsilon_t) = \sigma_1^2 \\ Cov(\epsilon_t, \epsilon_{t+h} = \rho_1 \end{cases} \quad and \quad \begin{cases} E[\nu_s] = \mu_2 \\ Var(\nu_s) = \sigma_2^2 \\ Cov(\nu_s, \nu_{s+h} = \rho_2 \end{cases}$$

where $\mu_1, \mu_2, \sigma_1, \sigma_2 \in \mathbb{R}$.

For $\{\epsilon_t + \nu_s\}$,

$$E[\epsilon_t + \nu_s] = E[\epsilon_t] + E[\nu_s] = \mu_1 + \mu_2 Var(\epsilon_t + \nu_s) = Var(\epsilon_t) + Var(\nu_s) + 2cov(\epsilon_t, \nu_s)$$

Since the process are non-correlated, then $cov(\epsilon_t, \nu_s) = 0$ and,

$$\begin{aligned} cov(\epsilon_t + \nu_s, \epsilon_{t+h} + \nu_{s+h}) &= cov(\epsilon_t, \epsilon_{t+h}) + cov(\epsilon_t, \nu_{s+g}) + cov(\nu_s, \epsilon_{t+h}) + cov(\nu_{s,s+h}) \\ &= cov(\epsilon_t, \epsilon_{t+h}) + cov(\nu_s, \nu_{s+h}) = \rho_1 + \rho_2 \end{aligned}$$

Then, $\{\epsilon_t + \nu_s\}$ is a stationary process.

Bibliography

[1] About OnionCat – https://www.onioncat.org/about-onioncat/.

[2] Bitcoin Developer Reference - https://bitcoin.org/en/developer-reference#p2p-network.

[3] Blockchain - https://www.blockchain.com/.

[4] Explorateur de blocs https://btc.com/.

[5] FIBRE Fast Internet Bitcoin Relay Engine - http://bitcoinfibre.org/.

[6] getblocktemplate - Bitcoin Wiki - https://en.bitcoin.it/wiki/Getblocktemplate.

[7] Global Bitcoin nodes distribution - https://bitnodes.earn.com/.

[8] P2p Network Guide - https://bitcoin.org/en/p2p-network-guide#misbehaving-nodes.

[9] Protocol documentation - Bitcoin Wiki - https://en.bitcoin.it/wiki/Protocol_documentation.

[10] Replace trickle nodes with per-node/message Poisson delays - https://github.com/bitcoin/bitcoin/commit/5400ef6bcb9d243b2b21697775aa6491115420f3.

[11] Bitcoin Core - https://github.com/bitcoin/bitcoin, June 2019. original-date: 2010-12-19T15:16:43Z.

[12] Novacoin - https://github.com/novacoin-project/novacoin, Apr. 2019. original-date: 2014-01-28T20:21:33Z.

[13] Aitzhan, N. Z., and Svetinovic, D. Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. *IEEE Transactions on Dependable and Secure Computing 15*, 5 (2016), 840–852.

[14] Back, A. Hashcash - A Denial of Service Counter-Measure. 10.

[15] Bashir, I. *Mastering blockchain*. Packt Publishing Ltd, 2017.

[16] Becker, G. Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis. 28.

[17] Biryukov, A., Khovratovich, D., and Pustogarov, I. Deanonymisation of clients in Bitcoin P2p network. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 15–29.

[18] Biryukov, A., and Pustogarov, I. Bitcoin over Tor isn't a good idea. In *2015 IEEE Symposium on Security and Privacy* (2015), IEEE, pp. 122–134.

[19] BitFury, G. Proof of Stake Versus Proof of Work. *White paper, Sep* (2015).

[20] Bordignon, T. Gnutella: Distributed System. 15.

[21] Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., and Wustrow, E. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 157–175.

[22] BOWDEN, R., KEELER, H. P., KRZESINSKI, A. E., AND TAYLOR, P. G. Block arrivals in the Bitcoin blockchain. *arXiv:1801.07447 [cs]* (Jan. 2018). arXiv: 1801.07447.

[23] BRAUNSTEIN, A., INGROSSO, A., AND MUNTONI, A. P. Network reconstruction from infection cascades. *Journal of the Royal Society Interface 16*, 151 (2019), 20180844.

[24] CACHIN, C., GUERRAOUI, R., AND RODRIGUES, L. *Introduction to Reliable and Secure Distributed Programming*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[25] CASTELLANOS, J. A. F., COLL-MAYOR, D., AND NOTHOLT, J. A. Cryptocurrency as guarantees of origin: Simulating a green certificate market with the Ethereum Blockchain. In *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)* (2017), IEEE, pp. 367–372.

[26] CHANDRA, T. D., AND TOUEG, S. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM (JACM) 43*, 2 (1996), 225–267.

[27] CHAUM, D. Blind signatures for untraceable payments. In *Advances in cryptology* (1983), Springer, pp. 199–203.

[28] CHAUM, D., FIAT, A., AND NAOR, M. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography* (1988), Springer, pp. 319–327.

[29] CHOUK, M. MASTER-SLAVE REPLICATION, FAILOVER AND DISTRIBUTED RECOVERY IN POSTGRESQL DATABASE. 133.

[30] CONG, L. W., HE, Z., AND LI, J. Decentralized Mining in Centralized Pools. 57.

[31] COULOURIS, J. D. G. Distributed Systems: Concepts and Design. *DISTRIBUTED SYSTEMS*, 1067.

[32] DAI, W. *B-Money-an anonymous, distributed electronic cash system*. Academic Press, 1998.

[33] DELGADO-SEGURA, S., PÉREZ-SOLÀ, C., HERRERA-JOANCOMARTÍ, J., NAVARRO-ARRIBAS, G., AND BORRELL, J. Cryptocurrency Networks: A New P2p Paradigm. *Mobile Information Systems 2018* (2018), 1–16.

[34] DENNING, P. J. The science of computing: The ARPANET after twenty years. *American Scientist 77*, 6 (1989), 530–534.

[35] DESHPANDE, V., BADIS, H., AND GEORGE, L. BTCmap: Mapping Bitcoin Peer-to-Peer Network Topology. In *2018 IFIP/IEEE International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN)* (2018), IEEE, pp. 1–6.

[36] DONET, J. A. D., AND HERRERA-JOANCOMARTÍ, J. Cryptocurrency P2p networks: a comparison analysis. *Actas de la XIV Reunión Espanõla de Criptologıa y Seguridad de la Información (RECSI 2016)* (2016), 423–428.

[37] DONET, J. A. D., PÉREZ-SOLA, C., AND HERRERA-JOANCOMARTÍ, J. The bitcoin P2p network. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 87–102.

[38] DONET DONET, J. A., PÉREZ-SOLÀ, C., AND HERRERA-JOANCOMARTÍ, J. The Bitcoin P2p Network. *Financial Cryptography and Data Security 8438* (2014), 87–102.

[39] DONNET, B. Introduction to Computer Security. 11.

[40] DUBOVITSKAYA, A., XU, Z., RYU, S., SCHUMACHER, M., AND WANG, F. How blockchain could empower ehealth: An application for radiation oncology. In *VLDB Workshop on Data Management and Analytics for Medicine and Healthcare* (2017), Springer, pp. 3–6.

[41] DUCHESNE, P., AND ROY, R. Robust tests for independence of two time series. *Statistica Sinica* (2003), 827–852.

[42] ET AL., S. Eventual Consistency. In *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, Boston, MA, 2009, pp. 1071–1072.

[43] EYAL, I., GENCER, A. E., SIRER, E. G., AND VAN RENESSE, R. Bitcoin-NG: A Scalable Blockchain Protocol. 22.

[44] EYAL, I., AND SIRER, E. G. Majority is not Enough: Bitcoin Mining is Vulnerable. 17.

[45] FANTI, G., AND VISWANATH, P. Deanonymization in the Bitcoin P2p Network. 10.

[46] FELD, S., SCHÖNFELD, M., AND WERNER, M. Analyzing the Deployment of Bitcoin's P2p Network under an AS-level Perspective. *Procedia Computer Science 32* (2014), 1121–1126.

[47] FELDMAN, M., PAPADIMITRIOU, C., CHUANG, J., AND STOICA, I. Free-riding and whitewashing in peer-to-peer systems. *IEEE Journal on selected areas in communications 24*, 5 (2006), 1010–1019.

[48] FINNEY, H. Rpow: Reusable proofs of work. *Internet: https://cryptome. org/rpow. htm* (2004).

[49] FISCHER, J., AND LYNCH, A. Impossibility of Distributed Consensuswith One Faulty Process. 9.

[50] FLEDER, M., KESTER, M. S., AND PILLAI, S. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657* (2015).

[51] FOX, A., AND BREWER, E. Harvest, yield, and scalable tolerant systems. In *Proceedings of the Seventh Workshop on Hot Topics in Operating Systems* (Rio Rico, AZ, USA, 1999), IEEE Comput. Soc, pp. 174–178.

[52] GARAY, J., KIAYIAS, A., AND LEONARDOS, N. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques* (2015), Springer, pp. 281–310.

[53] GARZIK, J. Public versus private blockchains. *BitFury Group, San Francisco, USA, White Paper 1* (2015).

[54] GERVAIS, A., CAPKUN, S., KARAME, G. O., AND GRUBER, D. On the privacy provisions of bloom filters in lightweight bitcoin clients. In *Proceedings of the 30th Annual Computer Security Applications Conference* (2014), ACM, pp. 326–335.

[55] GILBERT, S., AND LYNCH, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News 33*, 2 (2002), 51–59.

[56] GUERRAOUI, R., AND SCHIPER, A. The generic consensus service. *IEEE Transactions on Software Engineering 27*, 1 (2001), 29–41.

[57] HASLHOFER, B., KARL, R., AND FILTZ, E. O Bitcoin Where Art Thou? Insight into Large-Scale Transaction Graphs. In *SEMANTiCS (Posters, Demos, SuCCESS)* (2016).

[58] HEILMAN, E., KENDLER, A., ZOHAR, A., AND GOLDBERG, S. Eclipse attacks on bitcoin's peer-to-peer network. In *24th ${$USENIX$}$ Security Symposium (${$USENIX$}$ Security 15)* (2015), pp. 129–144.

[59] JAKOBSSON, M., AND JUELS, A. *Communications and Multimedia Security, chapter Proofs of Work and Bread Pudding Protocols*. Kluwer Academic Publishers, 1999.

[60] JALILI, M., AND PERC, M. Information cascades in complex networks. *Journal of Complex Networks 5*, 5 (2017), 665–693.

[61] JANI, S. An Overview of Ripple Technology & its Comparison with Bitcoin Technology. 6.

[62] JI, F., TANG, W., TAY, W. P., AND CHONG, E. K. Network Topology Inference Using Information Cascades with Limited Statistical Knowledge. *arXiv preprint arXiv:1706.09192* (2017).

[63] JUHÁSZ, P. L., STÉGER, J., KONDOR, D., AND VATTAY, G. A Bayesian Approach to Identify Bitcoin Users. *PLoS ONE 13*, 12 (Dec. 2018), e0207000. arXiv: 1612.06747.

[64] KATZ, J., MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of applied cryptography*. CRC press, 1996.

[65] KIM, S. K. MEASURING ETHEREUM'S PEER-TO-PEER NETWORK. 54.

[66] KING, S. Primecoin: Cryptocurrency with Prime Number Proof-of-Work. 6.

[67] KOCH, P. D., AND YANG, S.-S. A method for testing the independence of two time series that accounts for a potential pattern in the cross-correlation function. *Journal of the American Statistical Association 81*, 394 (1986), 533–544.

[68] KONSTANTINIDIS, I., SIAMINOS, G., TIMPLALEXIS, C., ZERVAS, P., PERISTERAS, V., AND DECKER, S. Blockchain for Business Applications: A Systematic Literature Review. In *Business Information Systems*, W. Abramowicz and A. Paschke, Eds., vol. 320. Springer International Publishing, Cham, 2018, pp. 384–399.

[69] KOSHY, P., KOSHY, D., AND MCDANIEL, P. An analysis of anonymity in bitcoin using p2p network traffic. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 469–485.

[70] LAMPORT, L., SHOSTAK, R., AND PEASE, M. The Byzantine Generals Problem. *ACM Transactions on Programming Languages and Systems 4*, 3 (July 1982), 382–401.

[71] LAW, L., SABETT, S., AND SOLINAS, J. How to make a mint: the cryptography of anonymous electronic cash. *Am. UL Rev. 46* (1996), 1131.

[72] LIU, P. T. S. Medical record system using blockchain, big data and tokenization. In *International conference on information and communications security* (2016), Springer, pp. 254–261.

[73] MALKHI, D., AND REITER, M. Byzantine quorum systems. *Distributed computing 11*, 4 (1998), 203–213.

[74] MALONE, D., AND O'DWYER, K. Bitcoin Mining and its Energy Footprint. In *25th IET Irish Signals & Systems Conference 2014 and 2014 China-Ireland International Conference on Information and Communities Technologies (ISSC 2014/CIICT 2014)* (Limerick, Ireland, 2014), Institution of Engineering and Technology, pp. 280–285.

[75] MEIKLEJOHN, S., POMAROLE, M., JORDAN, G., LEVCHENKO, K., MCCOY, D., VOELKER, G. M., AND SAVAGE, S. A fistful of bitcoins: characterizing payments among men with no names. In *Proceedings of the 2013 conference on Internet measurement conference* (2013), ACM, pp. 127–140.

[76] MERKLE, R. C. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques* (1987), Springer, pp. 369–378.

[77] MILLER, A., AND LAVIOLA, J. J. Anonymous Byzantine Consensus from Moderately-Hard Puzzles: A Model for Bitcoin. 7.

[78] MILLER, A., LITTON, J., PACHULSKI, A., GUPTA, N., LEVIN, D., SPRING, N., AND BHATTACHARJEE, B. Discovering bitcoin's public topology and influential nodes. *et al* (2015).

[79] MONACO, J. V. Identifying Bitcoin users by transaction behavior. I. A. Kakadiaris, A. Kumar, and W. J. Scheirer, Eds., p. 945704.

[80] MÜNSING, E., MATHER, J., AND MOURA, S. Blockchains for decentralized optimization of energy resources in microgrid networks. In *2017 IEEE conference on control technology and applications (CCTA)* (2017), IEEE, pp. 2164–2171.

[81] NAKAMOTO, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 9.

[82] NARAYANAN, A., BONNEAU, J., FELTEN, E., MILLER, A., AND GOLDFEDER, S. Bitcoin and Cryptocurrency Technologies. 308.

[83] NEUDECKER, T. Characterization of the Bitcoin Peer-to-Peer Network (2015-2018). 31.

[84] NEUDECKER, T., ANDELFINGER, P., AND HARTENSTEIN, H. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (2016), IEEE, pp. 358–367.

[85] PESARAN, M. H., AND TIMMERMANN, A. Testing dependence among serially correlated multicategory variables. *Journal of the American Statistical Association 104*, 485 (2009), 325–337.

[86] POESE, I., UHLIG, S., KAAFAR, M. A., DONNET, B., AND GUEYE, B. IP geolocation databases: Unreliable? *ACM SIGCOMM Computer Communication Review 41*, 2 (2011), 53–56.

[87] QING, S., OKAMOTO, T., AND ZHOU, J. *Information and communications security*. Springer, 2001.

[88] RABABAAH, H. DISTRIBUTED DATABASES FUNDAMENTALS AND RESEARCH. 16.

[89] RAUCHS, M., GLIDDEN, A., GORDON, B., PIETERS, G. C., RECANATINI, M., ROSTAND, F., VAGNEUR, K., AND ZHANG, B. Z. Distributed Ledger Technology Systems: A Conceptual Framework. *SSRN Electronic Journal* (2018).

[90] RIPEANU, M., IAMNITCHI, A., AND FOSTER, I. Mapping the gnutella network. *IEEE Internet Computing*, 1 (2002), 50–57.

[91] RON, D., AND SHAMIR, A. Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security* (2013), Springer, pp. 6–24.

[92] ROY, A. *Quantifying Relationships Between Two Time Series Data Sets*. PhD Thesis, North Dakota State University, 2016.

[93] SAI ANAND, R., AND MADHAVAN, C. An Online, Transferable E-Cash Payment System. In *Progress in Cryptology —INDOCRYPT 2000*, G. Goos, J. Hartmanis, J. van Leeuwen, B. Roy, and E. Okamoto, Eds., vol. 1977. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 93–103.

[94] SIKORSKI, J. J., HAUGHTON, J., AND KRAFT, M. Blockchain technology in the chemical industry: Machine-to-machine electricity market. *Applied Energy 195* (2017), 234–246.

[95] SOBTI, R., AND GEETHA, G. Cryptographic Hash Functions: A Review. 20.

[96] SOMMERS, J. A library for fast IP address lookup in Python. Contribute to jsommers/pytricia development by creating an account on GitHub, May 2019. original-date: 2012-08-14T19:52:18Z.

[97] STALLINGS, W. *Cryptography and network security: principles and practice*, 4th ed ed. Pearson/Prentice Hall, Upper Saddle River, N.J, 2006. OCLC: ocm63126393.

[98] STEEN, M. V., AND TANENBAUM, A. S. *Distributed systems*, third edition (version 3.01 (2017)) ed. Pearson Education, London, 2017. OCLC: 1006750554.

[99] STOCK, B., GÖBEL, J., ENGELBERTH, M., FREILING, F. C., AND HOLZ, T. Walowdac-analysis of a peer-to-peer botnet. In *2009 European Conference on Computer Network Defense* (2009), IEEE, pp. 13–20.

[100] STUTZBACH, D., AND REJAIE, R. Capturing accurate snapshots of the gnutella network. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (2006), IEEE, pp. 1–6.

[101] SUN, J., YAN, J., AND ZHANG, K. Z. K. Blockchain-based sharing services: What blockchain technology can contribute to smart cities. *Financial Innovation 2*, 1 (Dec. 2016).

[102] SZABO, N. Bit gold. *Website/Blog* (2008).

[103] SZÉKELY, G. J., RIZZO, M. L., AND BAKIROV, N. K. Measuring and testing dependence by correlation of distances. *The annals of statistics 35*, 6 (2007), 2769–2794.

[104] TAPSELL, J., AKRAM, R. N., AND MARKANTONAKIS, K. An evaluation of the security of the Bitcoin Peer-to- Peer Network. *arXiv:1805.10259 [cs]* (May 2018). arXiv: 1805.10259.

[105] TARKOMA, S., ROTHENBERG, C. E., AND LAGERSPETZ, E. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys & Tutorials 14*, 1 (2011), 131–155.

[106] TSCHORSCH, F., AND SCHEUERMANN, B. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Communications Surveys & Tutorials 18*, 3 (2016), 2084–2123.

[107] TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society 2*, 1 (1937), 230–265.

[108] VAN STEEN, M., AND TANENBAUM, A. S. A brief introduction to distributed systems. *Computing 98*, 10 (Oct. 2016), 967–1009.

[109] VASIN, P. BlackCoin's Proof-of-Stake Protocol v2. 2.

[110] VRUBLIAUSKAS, A. *Join/Leave Protocol for Structured Peer-to-Peer Networks*. Aalborg University. Department of Computer Science, 2003.

[111] WAHAB, A., AND MEMOOD, W. Survey of Consensus Protocols. 12.

[112] WANG, L., AND PUSTOGAROV, I. Towards Better Understanding of Bitcoin Unreachable Peers. *arXiv:1709.06837 [cs]* (Sept. 2017). arXiv: 1709.06837.

[113] YE, C., LI, G., CAI, H., GU, Y., AND FUKUDA, A. Analysis of Security in Blockchain: Case Study in 51%-Attack Detecting. In *2018 5th International Conference on Dependable Systems and Their Applications (DSA)* (Dalian, China, Sept. 2018), IEEE, pp. 15–24.

[114] ZANDER, S. On the accuracy of IP geolocation based on IP allocation data.

[115] ZHENG, Z., XIE, S., DAI, H., CHEN, X., AND WANG, H. An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In *2017 IEEE International Congress on Big Data (BigData Congress)* (Honolulu, HI, USA, June 2017), IEEE, pp. 557–564.

[116] ÖZSU, M. T., AND VALDURIEZ, P. *Principles of distributed database systems*, 3rd ed ed. Springer Science+Business Media, New York, 2011. OCLC: ocn706920112.